Tampereen yliopisto

Mikko Ilola

ÄLYKÄS NOPEUSMITTARI

Informaatioteknologian ja viestinnän tiedekunta
Diplomityö
Lokakuu 2019

TIIVISTELMÄ

Mikko Ilola: Älykäs nopeusmittari
Diplomityö
Tampereen yliopisto
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Lokakuu 2019

Nykyään on tarjolla useita erilaisia kaupallisia polkupyörään kiinnitettäviä laitteita, joiden toimintomäärä vaihtelee yksinkertaisesta nopeusmittarista täysiveriseen ajotietokoneeseen. Parhaimmissa laitteissa on suuri määrä erilaisia toimintoja, joiden toteutukset vaativat laajoja ohjelmistoja ja tehokkaita prosessoreita. Internetin keskustelupalstoilla on paljon kirjoituksia näiden laitteiden erilaisista toiminnallisista poikkeamista. Tämän diplomityön avulla tutustuttiin tällaisen laitteen suunnitteluprosessiin kokonaisuutena ja löydettiin syitä erilaisille poikkeamille niin komponenttien kuin ohjelmiston osalta.

Työn tarkoituksena oli suunnitella, rakentaa ja ohjelmoida polkupyörän älykäs nopeusmittari, joka käyttää GPS-paikannusta nopeuden ja kuljetun matkan määritykseen. Mittarissa on myös erillinen ilmanpaineanturi, jonka avulla määritetään nousu- ja laskumetrit. Harjoituksen aikana mitatut tiedot voidaan tallentaa SD-kortille. Kokonaisuuden osa-alueita olivat komponenttivalinnat, piirilevyn suunnittelu ja valmistus, kotelointi, ohjelmointi ja testaus. Työssä keskityttiin näiden osa-alueiden toteutukseen niin, että lopputuloksena syntyy virheettömästi ja sulavasti toimiva laite.

Työssä tarvittavaa laskentaa varten selvitettiin, miten kahden määritetyn GPS-sijainnin perusteella voidaan laskea sijaintien välinen matka tasavälisen lieriöprojektion avulla. Nousu- ja laskumetrien laskentaa varten selvitettiin miten korkeus saadaan laskettua yleisen paine-korkeus-kaavan avulla.

Elektroniikan suunnittelun osalta komponenttivalinnat suoritettiin siten, että laitteen virrankulutus ja fyysinen koko saataisiin mahdollisimman pieneksi. Laitteen ytimeksi valittiin AVR-sarjan mikroprosessori. Piirilevy suunniteltiin hyvien tapojen mukaisesti niin, että laitteen sisäiset ja ulkoiset häiriölähteet tulivat huomioiduiksi. Piirilevy valmistettiin kokonaisuudessaan käsin, mikä huomioitiin suunnittelussa ja mahdollisuuksien mukaan myös komponenttivalinnoissa. Laitteen kotelo 3D-mallinnettiin ja valmistettiin 3D-tulostamalla.

Suunnittelun suurin yksittäinen osa-alue oli ohjelmisto. Ohjelmisto toteutettiin C-ohjelmointikielellä ja kehitysympäristönä oli Atmel Studio. Laitteen mikroprosessori neuvottelee ohjelmiston avulla ulkoisten komponenttien kuten LCD-näytön, GPS-moduulin, ilmanpaineanturin, kalenteripiirin ja SD-kortin kanssa SPI-väylän avulla. Ohjelmistoa varten selvitettiin NMEA 0183 -standardin mukaisen lauserakenteen ominaisuudet. Laitteen mikroprosessori lukee GPS-moduulilta NMEA-lauseita ja tulkitsee ne ohjelmallisesti. Lisäksi tutustuttiin SD-kortin alustuksen ja FAT-tiedostojärjestelmän toteuttavaan FatFs-moduuliin. Ohjelmisto suunniteltiin ja toteutettiin modulaarisena ja helposti päivitettävänä. Myös ohjelmistossa huomioitiin laitteen virrankulutus minimoimalla käytetty prosessoriaika ja käyttämällä virransäästötiloja mahdollisimman paljon.

Laitteen käyttöliittymä suunniteltiin selkeäksi ja helppokäyttöiseksi. Laitteessa on trasflektiivinen LCD-näyttö, joka on luettavissa myös kirkkaassa valaistuksessa. Käyttöliittymää ohjataan neljän painonapin avulla. Laite antaa käyttäjälle toiminnoista palautetta myös äänimerkin avulla. Käyttöliittymän suunnittelussa ja toteutuksessa panostettiin nopeaan vasteeseen, jolloin laite reagoi käyttäjän toimintaan heti.

Työn suorittamisen aikana suoritettiin runsaasti erilaista testausta. Testaus suoritettiin kolmessa osassa siten, että erikseen tehtiin komponenttikohtainen testaus, ohjelmiston testaus ja kokonaisen laitteen toiminnallinen testaus. Jokaisen komponentin toiminta testattiin ohjelmistokehityksen edetessä huolellisesti yksitellen ennen laajempien toiminnallisuuksien toteutusta. Mahdollinen komponentin toiminnan virhe olisi voinut kertautua suunnittelun edetessä, jos sitä ei olisi huomattu. Ohjelmiston käyttöliittymän toiminta testattiin huolellisesti niin, että tavoiteltu nopea vaste saavutettiin. Myös valmiin laitteen testauksen käytettiin paljon aikaa, jotta lopulliset ominaisuudet saatiin toimimaan halutulla tavalla. Testauksen kaikissa vaiheissa löydettiin pienempiä ja suurempiä virheitä, joista suuri osa liittyi ohjelmistoon ja korjattiin ohjelmaa

muuttamalla. GPS-moduulin toiminnassa havaittiin testauksen aikana puutteita, joista jouduttiin olemaan yhteydessä valmistajaan asti. Puutteet saatiin korjattua lisäkomponenttien avulla.

Lopputuloksena työstä saatiin toimiva laite, joka toteuttaa kaikki suunnitellut ominaisuudet. Laitteessa on luotettava GPS-yhteys, jonka avulla laite näyttää nopeuden ja laskee kuljetun matkan. Laite mittaa myös käytettyä aikaa ja laskee keskinopeuden. Lisäksi laite mittaa ilmanpainetta ja laskee sen perusteella nousu- ja laskumetrit. Kaikki mitatut tiedot voidaan tallentaa SD-kortille tekstitiedostoon, josta niitä voidaan lukea esimerkiksi tietokoneella. Työn aikana laitteelle kirjattiin myös parannusehdotuksia jatkokehitystä ajatellen.

ABSTRACT

Mikko Ilola: Intelligent Speedometer
Master's thesis
Tampere University
Master's Degree Programme in Electrical Engineering
October 2019

Today there are many different commercial devices for use in bicycles with features ranging from simple speedometers to full bicycle computers. Top devices have many different features that require complex software and powerful processors. Many discussions about different discrepancies concerning these devices can be found in different internet forums. This thesis focuses on the design process of a bicycle computer as a whole. In the process, different reasons for these discrepancies were found in software as well as in electronic components.

The goal if this thesis was to design, implement and program an intelligent speedometer for bicycles. The device uses GPS to determine speed and distance travelled. The meter also uses a barometer to calculate ascent and decent in meters. All the collected information can be stored to an SD card. The thesis consists of different sections that include component selection, PCB design and manufacture, casing, programming and testing. The focus in every section was to build a device that works smoothly and is error-free.

For determining the distance between two measured GPS-locations equirectangular projection was used. For ascent and descent, the standard pressure altitude formula was used.

In designing the electronics, the components were selected based on current consumption and physical size. The goal was for the device to be energy efficient and as small as possible. The core component of the device was chosen from the AVR microprocessor family. PCB design was carried out using known good principles in such a way that internal and external sources of interference were taken into consideration. The PCB was manufactured by hand. This was taken into account in design and also in component selection were applicable. The case was 3D-modeled and 3D-printed.

The single largest part of the whole was software. Programming was carried out using C programming language and Atmel Studio. The microprocessor in the device talks to external components such as LCD display, GPS module, barometer, calendar chip and SD card via SPI bus. For use in programming the structure of NMEA 0183 sentences was familiarized with. The microprocessor reads these NMEA sentences from the GPS module and interprets them in software. FatFs module was used in SD card initialization and FAT file system. The software was designed to be modular and easy to update. Energy efficiency was also paid attention to in programming by minimizing processor time and utilizing sleep modes as much as possible.

User interface was designed to be simple and easy to use. The device uses a transflective LCD display that can be read even in direct sunlight. The user interface consists of four buttons. User gets audio feedback when using the buttons. Lots of effort was made to make the user interface fluent and fast responding.

During the course of the design process large amounts of testing was carried out. Testing was divided to three parts which were component testing, software testing and functional testing for the complete device. The correct function of every component was tested during software development before implementing larger scale functions. This was to ensure correct behavior. Errors at this stage could recur and be hard to find later. The user interface was tested thoroughly to ensure fluent function. A lot of time was used to test the complete device to so that all the features were sure to function correctly. In all parts of the testing smaller and larger discrepancies were found. Many on these were software based and corrected in programming. During the testing, such discrepancies in the function of the GPS module were found that the manufacturer had to be contacted. These discrepancies were corrected using additional components.

In the end a working device that fulfills all the designed features was produced. The device has a reliable GPS connection that is used to display speed and to calculate distance travelled. Time and average speed are also displayed. Using a barometer ascent and descent are displayed in meters. All the collected information can be saved to an SD card as a text file which can be read using a computer. During the design process many ideas to further improve the device were documented.

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

ALKUSANAT

Tämä diplomityö on tehty omana henkilökohtaisena projektina. Motivaationa työlle toimi oma laaja pyöräilyharrastukseni.

Kiitos työn ohjaajalle Karri Palovuorelle viimeistelyvinkeistä sekä avusta aiheen valinnassa ja työn kanssa alkuun pääsemisessä.

Kiitos myös vaimolleni Elina Ilolalle kannustuksesta työn suorittamisen aikana.

Ylöjärvellä, 1.10.2019

Mikko Ilola

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

AD-muunnin	Analog to Digital, muuttaa analogisen signaalin digitaaliseksi
ARM	Advanced RISC Maschines, 32-bittinen prosessoriarkkitehtuuri
ASCII	American Standard Code for Information Interchange, 7-bittinen tietokoneiden merkistö
BCD	Binary Coded Decimal, kymmenjärjestelmän lukujen binäärinen esitystapa
CS	Chip Select, piirin valintapinni
ESD	Electrostatic discharge, sähköstaattinen purkaus
FAT	File Allocation Table, Microsoftin kehittämä tiedostojärjestelmä
FFC	Flexible Flat Cable, joustava lattakaapeli
FIFO	First In First Out
FR-4	Flame Retardant, lasikuituipohjainen yleisesti käytetty piirilevymateriaali
GPS	Global Positioning System, maailmanlaajuiden satelliitteihin perustuva paikantamisjärjestelmä
IP67	Internatiol Protection, 6 = täysin pölytiivis, 7 = kestää hetkellisen upotuksen veteen
LCD	Liquid Crystal Display
NMEA	National Marine Electronics Association
OLED	Organic Ligth Emitting Diode
OTP	One Time Programmable
PLA	Polyaktidi, 3D-tulostuksessa käytettävä muotilaatu
SD	Secure Digital
SPI	Serial Peripheral Interface, sarjamuotoinen oheislaiteväylä
USART	Universal Synchronous/Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
ZIF	Zero Insertion Force, nollavoimaliitin

1. JOHDANTO

Nykyajan ihmiselle oman kehon toiminnasta saatavilla oleva tiedon määrä on viimeisen vuosikymmenen aikana kasvanut selvästi. Markkinoille on tullut erilaisia terveyttä ja suorituskykyä mittaavia laitteita, joiden avulla käyttäjä voi seurata kehonsa rasitusta levossa, arjessa ja urheilusuorituksen aikana. Yksi tällaisten laitteiden ryhmä on erilaiset polkupyörään kiinnitettävät nopeusmittarit, joissa toimintojen määrä vaihtelee yksinkertaisesta renkaan kierroksia laskevasta nopeusmittarista täysiveriseen ajotietokoneeseen, joka seuraa langattomasti sykettä, nopeutta, poljinnopeutta ja sijaintia ja osaa lähettää tietoja reaaliajassa matkapuhelimeen. Parhaimmat laitteet osaavat opastaa pyöräilijää valittua reittiä pitkin kartalla.

Uusien markkinoille lanseerattavien laitteiden haasteena on jatkuvasti lisääntyvien ominaisuuksien aiheuttama suuri ohjelmistokehityksen ja akkukapasiteetin tarve. Isommat näytöt ja nopeammat prosessorit kuluttavat enemmän virtaa, kun samalla laitteiden koko halutaan pitää pienenä, mikä asettaa fyysisiä rajoituksia akkujen koolle. Kasvavat ohjelmistot ovat vaikeampia hallita, mikä näkyy käyttäjille erilaisten virheiden muodossa. Jatkuvasti päivittyvät ohjelmistot ovat näissäkin laitteissa arkipäivää.

Tämän diplomityön tarkoituksena on suunnitella, rakentaa ja ohjelmoida polkupyörän nopeusmittari, joka sisältää älykkäitä ominaisuuksia, kuten matkan laskennan GPS-paikkatiedon perusteella ja tietojen tallennusmahdollisuuden SD-kortille. Laitetta voidaan pitää pienimuotoisena ajotietokoneena. Tämän työn avulla lukijan on mahdollista nähdä millaisia laitteen suunnittelussa kohdattavat haasteet konkreettisesti ovat. Tässä työssä esiteltyjen komponentteihin ja ohjelmistoon liittyvien ongelmatilanteiden aiheuttajat ja ratkaisut auttavat omalta osaltaan ymmärtämään monimutkaisempien ajotietokoneiden mahdollisen poikkeavan toiminnan taustaa.

Diplomityö on jaettu seitsemään lukuun. Luvussa 2 käsitellään teoreettista taustaa GPS-paikkatiedon perusteella suoritettavaa kuljetun matkan laskennasta. Tässä luvussa esitetään myös, miten mitatusta ilmanpaineesta saadaan laskettua korkeus ja tämän diplomityön kannalta tärkeä korkeuden muutos.

Luvussa 3 käydään läpi laitteen elektroniikan suunnittelu ja toteutus. Luvussa perustellaan laitteen komponenttivalinnat ja piirilevysuunnittelussa käytetyt periaatteet.

Lopuksi esitellään piirilevyn valmistus ja kokoonpano sekä käsin suoritettavaan kokoonpanoon liittyviä erityishuomioita.

Luvussa 4 esitellään laitteen kotelon suunnittelu 3D-mallinnusohjelmalla ja toteutus 3D-tulostimella. Luvussa esitellään myös kotelon kokoonpano ja työn edetessä kohdattuja erityishuomioita liittyen 3D-tulostuksen laatuun.

Luvussa 5 käsitellään laitteen ohjelmistoa. Ohjelmisto on suurin yksittäinen laitteen toimintaan vaikuttava osa-alue, joten sen rakenne ja käytetyt ratkaisut esitetään yksityiskohtaisesti. Luvussa esitellään GPS-moduulin käyttämä tietojen esitysmuoto NMEA ja erillinen vapaan lähdekoodin moduuli FatFs. Lisäksi luvussa käsitellään ohjelmiston perusrakennetta ja toimintaperiaatetta. Tärkeimmät yksittäiset funktiot ja toiminnalliset kokonaisuudet käsitellään erikseen. Lopuksi esitellään laitteen käyttöliittymä.

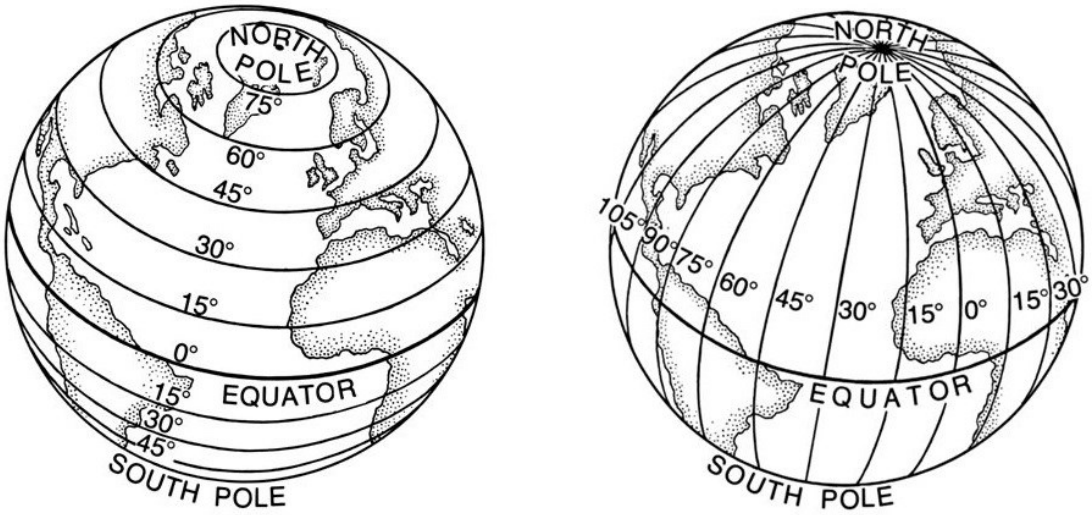
Luvussa 6 käydään läpi laitteen testaus. Luvussa käsitellään erikseen komponenttien testaus, ohjelmiston testaus ja toiminnallinen testaus. Jokaisesta osa-alueesta esitellään käytetyt testausmetodit, saadut testaustulokset ja suoritetut korjaustoimenpiteet.

Luvussa 7 tehdään yhteenveto työn kulusta, asetetuista tavoitteista ja niiden saavuttamisesta.

2. TEOREETTISTA TAUSTAA

2.1 Matkan laskenta GPS-koordinaateista

GPS-järjestelmässä paikka ilmoitetaan leveys- ja pituuskoordinaattien avulla. Koordinaatteja varten maapallo on jaettu leveys- ja pituuspiireihin. Kuvassa 2.1 vasemmalla on kuvattu leveyspiirit ja oikealla pituuspiirit. Myöhemmin esitettävää laskentaa varten pituuspiirien osalta voidaan huomioida, että piirien keskinäinen välimatka lyhenee, kun siirrytään kauemmaksi päiväntasaajasta. Leveyspiirien osalta näin ei tapahdu. Leveyspiiri 0° sijaitsee päiväntasaajalla ja pituuspiiri 0° kulkee Greenwichin kautta.



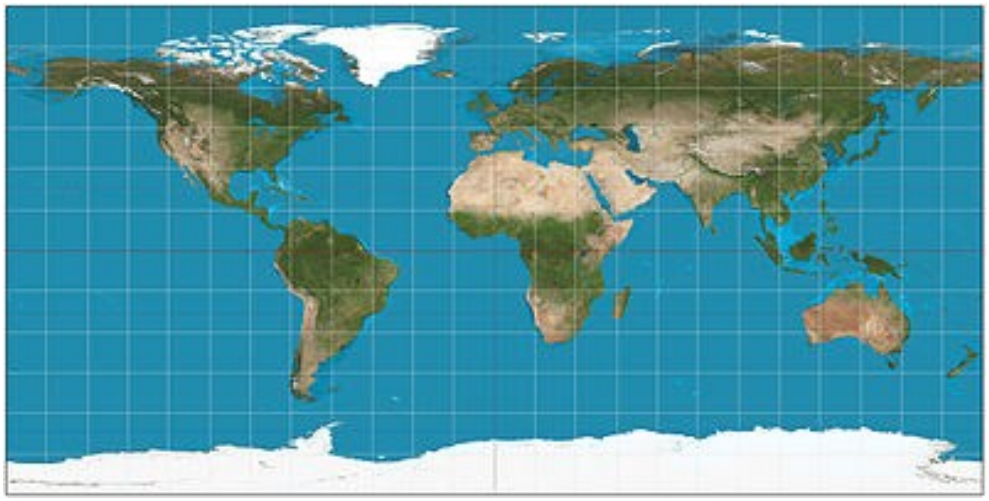
Kuva 2.1 *Maapallon jako leveys- ja pituuspiireihin.*

Leveyskoordinaatti on maapallon pinnalla olevan pisteen ja maapallon keskipisteen kautta piirretyn suoran viivan ja päiväntasaajan välinen kulma. Vastaavasti pituuskoordinaatti on maapallon pinnalla olevan pisteen ja maapallon keskipisteen kautta piirretyn suoran viivan ja Greenwichin kautta kulkevan pituuspiirin välinen kulma. Tällä esitystavalla maapallon oletetaan olevan pyöreä. Todellisuudessa maapallo on ellipsoidi ja tarkkuutta vaativassa paikan määrityksessä on mahdollista käyttää esitystapaa, jossa tämä on huomioitu. Tässä tapauksessa pallon pinnalla olevan pisteen kautta piirretty viiva ei kulje maapallon keskipisteen kautta, vaan sen vierestä.

Seuraavissa kappaleissa oletetaan maapallo pyöreäksi, joten tarkempaa esitystapaa ei käytetä.[20]

Maapallon pinnalla sijaitsevien kahden pisteen välisen etäisyyden tarkka määrittäminen vaatisi maapallon muodon huomioimista. Etenkin pidemmillä etäisyyksillä maapallon kaarevaa pintaa pitkin kuljettu matka on suurempi kuin kahden pisteen välinen suora etäisyys. Etäisyyden ollessa pieni, voidaan käyttää oletusta, jossa paikallisesti maapalloa pidetään tasaisena pisteiden välillä. Oletuksen aiheuttama virhe on hyvin pieni, kun liikutaan polkupyörällä esimerkiksi 25 km/h nopeudella ja paikka määritetään yhden sekunnin välein. Tässä tapauksessa yhden sekunnin aikana kuljetaan noin 6,9 metrin pituinen matka. Kun tätä verrataan myöhemmin laskennassa käytettävään maapallon keskisäteeseen, joka on 6371 km, voidaan todeta, että syntyvä virhe on häviävän pieni.

Edellä mainitun oletuksen perusteella suoritettavaa laskentaa varten pyöreä maapallo projisoidaan tasopinnaksi käyttäen tasavälistä lieriöprojektiota. Tässä projektiossa maapallon pinta muutetaan karteesiseen koordinaatistomuotoon niin, että koordinaatiston jokainen ruutu on saman kokoinen ja muotoinen ja niiden pinta-ala on sama. Päiväntasaajaa käytetään keskilinjana, jolloin ruudut ovat täydellisiä neliöitä. Projektion tarkkuus on parhaimmillaan pysty- ja vaakasuuntaisten keskilinjojen kohdalla ja huononee kun niistä loitonnutaan. Kuvassa 2.2 on esitetty koko maapallon pinta tasavälisenä lieriöprojektiona. [4]



Kuva 2.2 *Maapallo tasavälisenä lieriöprojektiona.*

Projektion jälkeen kahden pisteen välinen etäisyys voidaan laskea yksinkertaisesti Pythagoraan lauseen avulla:

$$d = \sqrt{\Delta x^2 + \Delta y^2} \quad (1)$$

Termit ∆x ja ∆y voidaan laskea kaavoilla:

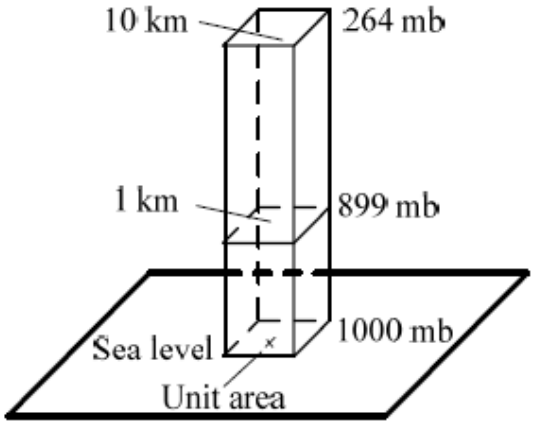
$$\Delta x = R\Delta\lambda \cos(\varphi) \quad (2)$$

$$\Delta y = R\Delta\varphi \quad (3)$$

Kaavoissa R tarkoittaa maapallon keskisädettä (6371 km), ∆φ on leveysasteiden erotus radiaaneina ja ∆λ on pituusasteiden erotus radiaaneina. Kosinitermin tehtävä kaavassa on kompensoida aiemmin mainittua pituuspiirien keskinäisen välimatkan muutosta, kun etäännytään päiväntasaajasta. Kulmana voidaan käyttää kumpaa tahansa leveysastetta radiaaneina. [14]

2.2 Korkeuden määritys ilmanpaineesta

Ilmanpaineen vaihtelu suhteessa korkeuteen on hyvin tunnettu ilmiö. Käyttämällä meren pintaa referenssitasona on ilmanpaineen avulla mahdollista määrittää korkeus. Ilmanpaine on määritelmän mukaan mittauspisteen yläpuolella sijaitsevan ilman paino pinta-alaa kohden. Mitä korkeammalla mittapiste sijaitsee, sitä pienempi on ilman paine (kuva 2.3).



Kuva 2.3 *Ilmanpaineen riippuvuus korkeudesta.*

Ilmakehän ominaisuudet eivät ole lämpötilaan suhteen vakioita, kun puhutaan korkeuden muutoksesta kymmenien kilometrien luokassa. Tässä työssä voidaan perustellusti keskittyä alimpaan ilmakehän alueeseen eli troposfääriin. Troposfäärin korkeus ei ole vakio, mutta se on pienimmilläänkin noin 6 km, joten suunniteltavan

mittarin käytön kannalta korkeusluokka on oikea. Troposfäärin alueella lämpötila laskee korkeuden lisääntyessä, jolloin ilmanpaineelle voidaan käyttää yleistä kaavaa: [11]

$$p = p_0 \left[1 - \frac{LH}{T_0}\right]^{\frac{g}{LR}} \quad (4)$$

Kun R ja g oletetaan vakioiksi, voidaan kaavasta edelleen ratkaista korkeus H:

$$H = \frac{T_0}{L}\left[1 - \left(\frac{p}{p_0}\right)^{\frac{LR}{g}}\right] \quad (5)$$

Kaavassa käytetyt merkinnät ovat:

- $T_{0,}$ vakiolämpötila meren pinnan tasossa 288,15 K
- L, nopeuden muutosnopeus, joka troposfäärin alueella on -6,50 K/km
- p, mitattu ilmanpaine
- p_0, ilmanpaine meren pinnan tasossa 101325 Pa
- R, kaasuvakio ilmalle 287,05287 $m^2/(K \cdot s^2)$
- g, putouskiihtyvyys 9,81 m/s^2

Kun arvot sijoitetaan kaavaan, saadaan se ohjelmoinnissa käyttöä varten yksinkertaistettua muotoon: [18]

$$H = -44330{,}8\left[\left(\frac{p}{101325}\right)^{0{,}190263102} - 1\right] \quad (6)$$

Kaavalla saadaan laskettua korkeus mitatusta ilmanpaineesta. Ilmanpaineen mittaustulokseen vaikuttaa kuitenkin vallitseva säätila, jolloin samalla korkeudella ilmanpaine saattaa vaihdella. Tässä työssä rakennettava laite keskittyy mittaamaan korkeuden muutosta, eikä absoluuttista korkeuden oikeaa arvoa esitetä käyttäjälle tai käytetä laskennassa. Mittausväli ilmanpaineelle laitteessa on yksi sekunti ja laskennassa oletetaan, että ilmanpaineen perusteella laskettua korkeuden muutosta voidaan tällä lyhyellä aikavälillä pitää riittävän tarkkana. Myös lämpötila vaikuttaa ilmanpaineeseen. Lämpötilan aiheuttama virhe on mahdollista huomioida laskennallisesti. Tarkempi laskentamalli on esitetty kappaleessa 5.3.2.

3. ELEKTRONIIKAN SUUNNITTELU

Laitteen elektroniikan suunnittelussa käytettiin Mentor Graphics:n PADS -ohjelmistoa. Tämä ohjelmisto sisältää erilliset moduulit piirikaavion piirtämiseen (PADS Logic), piirilevyn suunnitteluun (PADS Layout) ja piirilevyn reititykseen (PADS Router). Ohjelmiston valinta perustui laajaan kokemukseen PADS:n käytöstä opintojen aikana, ja koska PADS tarjoaa kaikki tarvittavat työkalut tässä työssä suunniteltavan laitteen toteuttamiseen, ei tarvetta muiden ohjelmistojen tarkempaan tutkimiseen ollut. Kilpaileva ohjelmisto PADS:lle olisi ollut esimerkiksi Altium Designer.

Suunniteltava laite on toiminnaltaan täysin digitaalinen, joten erilliselle analogiselle simulaattoriohjelmistolle ei ollut tarvetta. Laitteen ainoa analoginen sisääntulo on akkujännite, joka kytkentään mikrokontrolleriin yksinkertaisen vastusjaon kautta. Kytkentä on niin yksinkertainen, että sen toimintaa ei ollut tarvetta simuloida.

Piirilevyllä kulkevien signaalien laatua olisi ollut mahdollista mallintaa erillisillä ohjelmistoilla, mutta käytössä olevat taajuudet ovat kohtuullisen matalia, joten ohjelmiston käyttöä ei katsottu tarpeelliseksi. Erilaiset digitaalisen väyläliikenteen ongelmat kuten heijastukset tai ylikuuluminen pyrittiin välttämään käyttämällä piirilevysuunnittelussa hyvän tavan mukaisia periaatteita ja tarvittaessa poistamaan häiriöitä yksinkertaisilla komponenteilla.

3.1 Komponenttivalinnat

Laitteen toiminta rakentuu periaatteelle, jossa kaikkia toimintoja ohjaa mikrokontrolleri. Kaikki ulkoiset piirit keskustelevat mikrokontrollerin kanssa sarjamuotoisen SPI-tiedonsiirron avulla. Mikrokontrolleri on laitteen tärkein yksittäinen komponentti, joten sen valinta suoritettiin ensimmäisenä. Nykyään on yhä laajemmin saatavilla ARM -pohjaisia, 32 -bittisiä korkean kellotaajuuden mikrokontrollereita, jotka tarjoavat hyvää nopeutta yhdistettynä pieneen virrankulutukseen. Suunniteltavan laitteen ohjelmisto ei ole kuitenkaan suurta prosessoritehoa vaativa, koska suurin osa ohjelmiston toiminnasta on väyläliikenteen hallintaa ja ulkoisten komponenttien viestien tulkintaa. Tästä syystä mikrokontrollerivalinta kohdistui Atmelin AVR AtMega -sarjaan. Valintaa tuki helposti saatavilla ollut ohjelmiston kehitysalusta (Atmel Studio) ja ennestään olemassa ollut AVR -yhteensopiva ohjelmointilaite.

Mikrokontrollerin tärkein tarkempi valintakriteeri oli käyttöjännite. Suunniteltava laite toimii akkujännitteellä, joten virrankulutus oli tärkeää saada mahdollisimman pieneksi.

Tätä silmällä pitäen käyttöjännitteeksi valittiin 3,3 V. Valittavan mikrokontrollerin piti pystyä toimimaan tällä jännitteellä riittävällä kellotaajuudella. Mikrokontrolleriksi valittiin AtMega 1281, joka pystyy toimimaan 3,3 V jännitteellä noin 10 MHz kellotaajuudella, jolloin käytännössä tällä jännitteellä kontrollerin sisäistä 8 MHz resonaattoria voitiin käyttää täydellä nopeudella. Sisäinen resonaattori ei ole taajuudeltaan tarkka, mutta käytössä oleva SPI-väylä ei ole riippuvainen kellosignaalin tarkkuudesta, koska dataliikenne synkronoidaan ohjaavan laitteen kellosignaalilla. Lisäksi ohjelman ei katsottu olevan niin laaja, että laskentateho loppuisi kesken. AtMega 1281 sisältää 128 KB flash- ja 8 KB SRAM-muistia. Muistia arveltiin todennäköisesti olevan liikaa, mutta mahdollista sarjatuotantoa ajatellen kontrolleri on mahdollista vaihtaa suoraan pienempään ja halvempaan malliin. [2]

Laitteen käytön kannalta näkyvin komponentti on näyttö, jolle valintakriteereiksi muodostuivat luettavuus, SPI-liityntä ja käyttöjännite. Laitteen pääasiallinen käyttöpaikka on ulkona ja usein auringonpaisteessa, joten näytön tulee olla luettavissa myös kirkkaassa valaistuksessa. Nykyään on saatavilla pieniäkin OLED-tekniikkaan perustuvia näyttöjä, jotka ovat kirkkaita ja luettavuus ulkona voisi olla hyvällä tasolla. Tämä tarkoittaisi kuitenkin, että näyttöä pitäisi ajaa jatkuvasti maksimikirkkaudella, mikä aiheuttaisi suuren virrankulutuksen. Transflektiivinen LCD-näyttö sen sijaan toimii kirkkaassa valaistuksessa täysin ilman taustavaloa, koska näytön pikselit valaistaan heijastamalla näytön ulkopuolelta tulevaa valoa takaisin pikselin läpi. Kirkkaampi valaistus aiheuttaa automaattisesti voimakkaamman valaistuksen, joten näyttö pysyy luettavana. Näytöksi valittiin Midas MCCOG128064B12W-FPTLW, joka on 128 x 64 pikselin graafinen, transflektiivinen LCD-näyttö. Näyttö toimii 3,3 V jännitteellä, tukee ohjausta SPI-väylän kautta ja sisältää sisäänrakennetun valkoisen taustavalon. Näytön näkyvä osuus on kooltaan 50,50 mm x 31,00 mm. Näytön koko on käytön kannalta hyvin sopiva.

GPS-piiri on laitteen tärkein erillinen tietolähde. Piirilevylle tai koteloon ei ollut tarkoitus suunnitella erillistä antennia, joten GPS-piirin tuli olla täysin integroitu moduuli. Käytössä oleva SPI-väylä rajaa mahdollisuuksia voimakkaasti, koska GPS -moduuleissa on yleisemmin käytössä USART-väylä. Aikaisemmin valitulla mikrokontrollerilla USART-väylän käyttö olisi täysin mahdollista, mutta laite on suunniteltu pelkästään SPI-väylää käyttäväksi, joten GPS-moduuliksi valikoitui Maestron A2235-H, joka toimii 3,3 V jännitteellä, tukee SPI-väylää ja sisältää sisäänrakennetun antennin. Moduuli on myös ulkoisilta mitoiltaan riittävän pieni (17,78 mm x 16,51 mm x 7,11 mm).

Erillisen lämpötila- ja ilmanpaineanturin osalta valintakriteerit vastasivat muita komponentteja. Anturiksi valittiin Measurement Specialties:n MS5607-02BA03, joka

toimii 3,3 V jännitteellä ja tukee SPI-väylää. Anturi pystyy mittaamaan ilmapainetta väliltä 10 – 1200 mbar maksitarkkuuden ollessa 0,024 mbar. Lämpötilaa anturi pystyy mittaamaan väliltä -40 – +85 °C tarkkuudella 0,01 °C. Anturi sisältää 24-bittisen AD-muuntimen, joten paine- ja lämpötilatieto saadaan suoraan digitaalisessa muodossa. Anturi on tarkoitettu kannettaviin laitteisiin ja on kooltaan erittäin pieni (5.0 x 3.0 mm). [17]

Tarkkaa ajanottoa varten mikrokontrolleriin olisi ollut mahdollista lisätä ulkoinen 32768 Hz kide reaaliaikakelloa varten. Virrankulutuksen minimoimiseksi mikrokontrolleri oli kuitenkin tarkoitus sammuttaa käytön aikana mahdollisimman pitkäksi ajaksi, joten ajanotto päätettiin hoitaa erillisellä ulkoisella erittäin vähän virtaa kuluttavalla kalenteripiirillä. Useat tähän tarkoitukseen suunnitellut piirit tarvitsivat toimiakseen erillisen ulkoisen kiteen. Piirilevytilan säästöä ajatellen valinta kohdistui NXP Semiconductors:n PCF2129-piiriin, joka sisältää sisäänrakennetun lämpötilakompensoidun kiteen, eikä vaadi toimiakseen ulkoisia komponentteja. Piiri täyttää käyttöjännite- ja väyläliityntävaatimukset ja sisältää lisäksi kaksi erillistä ohjelmoitavaa ulostuloa, joilla mikrokontrollerin toimintaa voidaan tahdistaa tarkasti.

Laitteen käyttöliittymä toimii neljällä erillisellä painonapilla. Napeiksi valittiin piirilevyn pintaan kiinnitettävät C&K Switches:n KSC9:t, jotka ovat IP67-suojattuja ja niihin on mahdollista kiinnittää erillinen hattu. Painonappien värähtely kytkentähetkellä olisi ollut mahdollista suodattaa ohjelmallisesti, mutta suodatusta varten on olemassa myös erillisiä suodatuspiirejä (*engl. debounce)*. Erillistä piiriä käyttämällä ohjelmiston ei tarvitse odottaa värähtelyn tasoittumista ja lisäksi piiri tarjoaa ESD -suojauksen napin ja mikrokontrollerin välille. Laitteen todennäköinen käyttäjä on urheilija, joka on usein pukeutunut keinokuituisiin vaatteisiin, jolloin staattisen sähkön aiheuttaman purkauksen vaara käyttäjän sormesta painonappiin on olemassa. Piiriksi valittiin Maxim Integrated:n MAX6817, joka täyttää käyttöjännitevaatimuksen, ja jonka luotettavasta toiminnasta oli aiempaa kokemusta harjoitustyönä rakennetussa lentokoneen moottorin-valvontamittarissa.

Äänen tuottoa varten laitteeseen tarvittiin summeri. Ääntä haluttiin tuottaa mahdollisimman pienellä komponenttimäärällä, mikä tarkoitti käytännössä suoraa ohjausta mikrokontrollerista. Summeriksi valittiin Multicomp:n pietsoelektrinen MCABT-458-RC, joka tuottaa 3 Vp-p -jännitteellä ja 1 mA virralla vähintään 75 dB äänen 10 cm

etäisyydelle, kun taajuus on 4 kHz. Summeri on kevyt ja kooltaan 12.0 mm x 12.2 mm x 3 mm.

Erilaisia kytkentätilanteita varten tarvittiin sekä N- että P-kanavaisia MOSFET-transistoreita. Transistorien tuli olla täysin johtavassa tilassa 3,3 V jännitteellä. Tarvittavat virrat ovat pieniä (alle 100 mA), joten tämä ei muodostunut rajoitteeksi. Transistoreiksi valittiin helposti saatavilla olevat ON Semiconductor:n NTR4003N ja NTR4502P.

Laitteen teholähteenä toimii 1200 mAh litiumpolymeeriakku, joka on liitetty erilliseen lataus- ja valvontamoduuliin (Wemos 18650 Battery Shield V3). Moduulia on muokattu muuttamalla alkuperäisen 18650-akun tilalle litteä, paremmin koteloon mahtuva akku. Moduuli tuottaa 5 V käyttöjännitettä, jota tarvitaan LCD -näytön taustavaloon ja josta reguloidaan muiden piirien tarvitsema 3,3 V käyttöjännite. Regulaattoriksi valittiin Microchip:n TC1262, jonka pudotusjännite (*engl. drop-out voltage*) on 390 mA kuormalla pieni 200 mV ja pienemmällä 100 mA kuormalla 60 mV. Regulaattorin maksimikuorma on 500 mA, joka olisi muiden komponenttien osalta voimakkaasti ylimitoitettu, mutta koska laitteeseen on mahdollista kytkeä SD kortti, muuttui virrankulutuksen suunnittelu selvästi. SD-kortille ei ole mahdollista määrittää yksittäistä ainoaa maksimivirtaa, koska eri valmistajien korttien välillä virrankulutukset saattavat erota toisistaan yli 100 mA. Tyypilliset arvot eri valmistajien datalehdillä vaihtelevat välillä 50 mA – 150 mA, mutta vanhemmilla korteilla suuremmatkin arvot ovat mahdollisia. Alla oleva taulukko 3.1 kertoo eri komponenttien maksimivirrankulutukset suurimmasta pienimpään lajiteltuna. SD-kortti on selvästi eniten kuluttava komponentti ja regulaattori mitoitettiin niin, että enitenkään virtaa kuluttava kortti ei tule aiheuttamaan ongelmia laitteen toiminnalle.

Taulukko 3.1 *Yhteenlaskettu virrankulutus.*

Muistikortti	300	mA
GPS-moduuli	42	mA
Taustavalo	40	mA
Mikrokontrolleri	20	mA
Paineanturi	1,4	mA
Summeri	1	mA
Kalenteri	1	mA
LCD-näyttö	0,1	mA
Debounce	0,02	mA
Yhteensä	405,5	mA

3.2 Kytkentäkaavio

Kytkentäkaavion piirtämisessä tarkoituksena on esittää erilaisten komponenttien väliset yhteydet toisiinsa. Suunnittelussa ensimmäisenä sijoitettiin komponentit loogisiksi kokonaisuuksiksi, jotta kaaviosta tulisi mahdollisimman selkeä ja helppolukuinen. Tässä tapauksessa kaikki kytkennät mahtuivat yhdelle sivulle, joten kokonaisuutta ei jaettu turhaan osiin. Kytkentäkaavio on luettavissa liitteessä 1.

Passiivikomponentteja lukuun ottamatta valitut komponentit olivat sellaisia, joita ei löytynyt ohjelmiston kirjastoista valmiina. Jokaiselle komponentille piirrettiin oma piirilevykaavion kuva, jossa esitettiin komponentin nastat numeroituna ja nimettynä. Komponenttien datalehtien avulla selvitettiin jokaisen nastan oikea käyttötarkoitus ja kytkentätapa. Myös komponenttien tarvitsemien ulkoisten passiivikomponenttien kuten erilaisten vastusten ja kondensaattorien tarve selvitettiin datalehdistä. Hyvänä esimerkkinä oli LCD-näytön kytkentä, johon kuuluu 10 ulkoista kondensaattoria. Näytön ohjainpiirissä on integroitu hakkuri, joka muodostaa näytön tarvitsemat käyttöjännitteet ulkoisten kondensaattoreiden avulla. Kondensaattorien kytkentä valittiin käyttöjännitteen mukaan. Jännitteen ollessa 3,3 V tarvittiin kytkentä, joka nostaa jännitteen nelinkertaiseksi [16].

Kytkentä sisältää paljon erilaisia on-off -tyyppisiä ohjauksia, joista osa kulkee ulkoisten transistoreiden kautta ja osa suoraan komponenteille. Kytkennässä kaikkiin tällaisiin ohjauksiin kytkettiin joko ylös- tai alasvetovastus riippuen siitä kummassa tilassa komponentin nastan haluttiin pysyvän ilman ohjausta. Tällä varmistettiin, että esimerkiksi SPI-väylään liitetyt laitteet pysyivät hallitusti passiivisina käynnistyksen yhteydessä. Vastaavasti LCD-näytön taustavalo pysyi hallitusti pois päältä. Myös SPI-väylän dataväyliin lisättiin ylösvetovastukset. SPI-väylän toiminnan kannalta vastukset eivät olleet tarpeellisia, mutta niillä haluttiin estää häiriöiden kytkeytyminen väylään tilanteessa, jossa SD-korttia ei ole asennettu. Vastaavasti myös SPI-väylän kellosignaali on alttiina häiriöille, mutta koska osa piireistä vaatii kellosignaalin olevan levossa ylhäällä ja osa alhaalla ei vastusten asentaminen ollut mahdollista. Mikrokontrolleri ajaa kellosignaalia kuitenkin voimakkaasti, joten alttius häiriöille on pieni.

Regulaattorin sisään- ja ulostuloihin kytkettiin datalehden suosittelemat 1µF kondensaattorit ja kaikkien piirien käyttöjännitteet suodatettiin paikallisesti 100 nF -kondensaattoreilla. Lisäksi mikrokontrollerin analogiapuolen jännitesyöttöön kytkettiin datalehden suosittelema alipäästösuodatin, koska kontrollerin sisäistä AD -muunninta käytettiin akkujännitteen tarkkailuun [2]. Laitteen akkujännite mitataan passiivisen vastusjaon avulla. Vastukset mitoitettiin niin, että akkujännitteen ollessa 5 V, näkyy

mikrokontrollerin sisääntulossa 3,3 V. Käytännössä akkujännite vaihtelee välillä 3,3 V – 4,2 V, mutta kytkennällä estettiin ylijännite kontrollerin sisääntulossa. Akkujännite kytketään vastusjakoon transistorin avulla vain mittauksen ajaksi, jotta vastusten kautta ei kulje turhaan virtaa.

Pietsoelektrisen summerin rinnalle kytkettiin vastus, jonka kautta summeriin kerääntynyt varaus pääsee purkautumaan, kun jännitettä ei ole kytketty. Summeria ohjataan kanttiaallolla transistorin kautta. Ilman vastusta ääni voisi jäädä hiljaiseksi tai ei kuuluisi ollenkaan.

3.3 Piirilevyn suunnittelu

Prototyypin piirilevyt syövytettiin ja juotettiin käsin, joten piirilevysuunnittelun lähtökohtana oli kaksikerroksinen levy. Suunnittelussa otettiin kuitenkin huomioon mahdollinen tarve siirtyä nelikerroksiseen levyyn. Käytännössä tämä toteutettiin niin, että levyn yläpuolella suurin osa tarvittavista hyppylangoista oli käyttöjännitevetoja, jolloin käyttöjännite olisi myöhemmin helppo siirtää omaan reitityskerrokseensa.

Piirilevyn suunnittelu aloitettiin komponenttien sijoittelulla. Sijoittelu on erityisen tärkeää tehdä huolellisesti, jotta piirilevyn reitit ykset pystytään pitämään mahdollisimman lyhyinä ja signaalien kulkureittejä pystytään hallitsemaan niin, että ylimääräisiä silmukoita ei pääse syntymään. Silmukat säteilevät häiriöitä sekä ulospäin että mahdollistavat ulkoisten häiriöiden kytkeytymistä. [6]

Suunniteltavan laitteen osasijoittelussa hallitsevina tekijöinä ovat LCD -näyttö, GPS -moduuli ja painonapit. Käytännössä edellä mainitut komponentit sijoitettiin toiselle ja loput komponentit toiselle puolelle levyä. LCD -näyttö kiinnittyy lähes kiinni piirilevyyn pintaan, joten sen alle ei ollut mahdollista sijoittaa komponentteja. Painonappien on käyttämisen kannalta oltava samalla puolella levyä kuin LCD -näyttö. GPS -moduulin antenni pitää osoittaa kohti taivasta parasta suorituskykyä ajatellen. Äänen kuulumisen varmistamiseksi myös summeri sijoitettiin samalle puolelle levyä kuin LCD -näyttö.

Edellä mainittujen komponenttien koko ja sijoittelu määrittivät tarvittavan levyn koon käytännössä kokonaan ja levyn kooksi muodostui 80,0 mm x 65,0 mm. Pienempi levyn koko olisi laitteen käyttötarkoitus huomioon ottaen ollut parempi, mutta valittuja ominaisuuksia ei pystytty toteuttamaan komponenttien fyysisen koon takia pienemmällä levyllä. Pienempi koko olisi vaatinut vähintään SD-kortin jättämisen pois, erillisen GPS-antennin ja pienemmän summerin.

Jäljelle jäävät komponentit ja virta- ja ohjelmointiliittimet sijoitettiin toiselle puolelle levyä niin, että SPI-väylä saatiin kulkemaan levyllä mahdollisimman lyhyttä ja yksinkertaista

reittiä pitkin. Suodatuskondensaattorit sijoitettiin mahdollisimman lähelle komponentteja niin, että käyttöjännite reititettiin kondensaattoriin ja siitä edelleen komponentin nastalle (kuva 3.1). Näin kondensaattori toimii suunnitellulla tavalla ja suodattaa häiriöitä.

Kuva 3.1. *Suodatuskondensaattorin sijoittelu. [1]*

Suunniteltava laite oli akkujännitteen mittausta lukuun ottamatta täysin digitaalinen, joten maadoitusmenetelmäksi valittiin maataso, joka sijoitettiin levyn toiselle puolelle ja pyrittiin pitämään mahdollisimman yhtenäisenä. Maatason avulla varmistettiin signaaleille aina mahdollisimman lyhyt paluureitti lähtöpisteeseen. Komponenttien kytkentä maatasoon tehtiin lyhintä mahdollisinta reittiä käyttäen läpivientien avulla. Läpivientien määrä nousi suureksi, joten maatason puolelle niille ei käytetty juottamista helpottavia padeja, vaan läpiviennit juotettiin suoraan maatasoon. Näin maatasosta ei tullut rikkonainen. Koneellisessa levyn valmistuksessa vastaavaa ongelmaa ei olisi muodostunut. Komponenttien sijoittelulla voitiin myös vaikuttaa virtojen kulkureitteihin ja komponentit sijoitettiin niin, että toisen komponentin paluuvirrat eivät kulje esimerkiksi toisen komponentin alta, jolloin virtojen kulkuun ei ollut tarpeellista puuttua maatasossa. Regulaattori sijoitettiin levylle niin, että sillä on käytössä mahdollisimman häiriötön maa. Akkujännitteen liitin sijoitettiin regulaattorin viereen ja tarvittavat lyhyet vedot suunniteltiin niin, että muodostuvien silmukoiden pinta-ala minimoitui. Myös regulaattorin kannalta muut komponentit sijoitettiin niin, että paluuvirrat eivät häiritse regulaattorin maata. [13]

Akkujännitteen mittaus suoritetaan analogisesti ja muunnetaan digitaaliseksi mikrokontrollerin sisäisellä AD -muuntimella. Mittauksen tarkkuustarve ei ole suuri, koska käytännössä mitataan 0,1 V eroja, joten erillistä maadoitusta analogiapuolelle ei järjestetty.

SPI-väylällä käytettävä taajuus on 4 MHz. Käytännössä taajuus on niin pieni, että sen ei todennäköisesti ajateltu aiheuttavan häiriöitä, kun väylän linjat suunniteltiin hyvien periaatteiden mukaisesti. Käytännössä linjat kuljetettiin mahdollisimman kaukana toisistaan, pidettiin lyhyinä ja minimoitiin silmukat. Vaikka SPI-väylän kellotaajuus ei todennäköisesti aiheuta ongelmia, huomioitiin suunnittelussa kuitenkin kanttiaallon nousunopeus. Nopea nousu eli lyhyt nousuaika aiheuttaa aallon reunalla värähtelyä,

joka saattaa aiheuttaa toimintahäiriöitä. Todennäköistä on, että tämän kokoisessa laitteessa nousuaikakaan ei tekisi laitetta toimimattomaksi, mutta jokaisen laitteen lähettävään päähän lisättiin kuitenkin sarjavastus hidastamaan signaalin nousua. Vastus toimii yhdessä signaalilinjan parasiittisen kapasitanssin kanssa alipäästösuodattimena. [5]

Piirilevyn pienimmät yksityiskohdat ovat kooltaan 0,2 mm (8 mils). Jotta yksityiskohdat säilyvät levyllä valmistuksen aikana on syövytyksessä poistuvan kuparin määrä tärkeää huomioida jo suunnitteluvaiheessa. Jos levyllä on suuria alueita, joilta kupari syöpyy pois, kiihdyttää suuri kuparin määrä reaktiota ja kuumentaa syövytysnestettä, mikä aiheuttaa helposti pienien yksityiskohtien liiallista syöpymistä. Tämän välttämiseksi levy suunniteltiin yleisesti niin, että syövytettävän kuparin määrä oli mahdollisimman pieni ja lisäksi alueet, joissa oli pieniä yksityiskohtia, huomioitiin jättämällä niiden läheisyyteen kuparialueita. Jätetyt kuparialueet liitettiin useista kohdista maatasoon, jotta niihin ei kytkeydy häiriöitä [13]. Laitteen ylä- ja alapuolen valotusmaskit ja osasijoittelukuvat sekä porauskaavio on kuvattu liitteistä 2 – 5.

Laitteessa ei ole voimakkaasti lämpöä tuottavia komponentteja, joten lämmön siirtymiseen ja haihtumiseen ei kiinnitetty suunnittelusta huomioita. Suuremmassa kuormitustilanteessa regulaattorissa mahdollisesti syntyvä lämpö pääsee siirtymään tehokkaasti maatasoon useiden läpivientien kautta.

3.4 Piirilevyn valmistus

Piirilevy valmistettiin käsityönä käyttäen 1,6 mm vahvuista FR-4 piirilevyä, jossa kuparikerroksen vahvuus oli 35 µm. Valmistuksen ensimmäinen vaihe oli valotusmaskien tulostaminen kalvolle. Tulostimen laatu vaikuttaa lopputulokseen, mutta käytettävissä olleen lasertulostimen tarkkuus oli kokemuksen perusteella niin hyvä, että yksityiskohtien kanssa ei odotettu tulevan ongelmia. Tulostusjälki ei yleensä kuitenkaan ole täysin tasaista vaan mustille alueille saattaa jäädä yksittäisiä pisteitä, joista valo pääsee kulkemaan läpi. Käyttämällä kahta samanlaista kalvoa päällekkäin tämä ongelma poistuu, koska pisteet muodostuvat yleensä eri tulostuskerroilla eri kohtiin. Molempien puolien kahdennetut maskit asetettiin kohdakkain ja piirilevy laitettiin oikealle kohdalleen maskien väliin. Piirilevyn pinnassa olevaa valoherkkää positiivista fotoresistiä valotettiin erillisellä laitteella. Yleinen valotusaika on 45 s – 60 s riippuen käytettävistä kalvoista. Kahdella kalvolla 60 s toimi vielä hyvin, eikä ylimääräistä valotusta tapahtunut.

Valotettu piirilevy kehitettiin lipeäliuoksessa, jolloin valottuneet osiot liukenivat pois. Kehitysajaksi riitti 5 s – 10 s levyn puolta kohden. Liiallista kehittämistä kannattaa välttää,

jos levyllä on pieniä yksityiskohtia. Kehityksen jälkeen levy on hyvä huuhdella runsaalla vedellä ennen syövytystä.

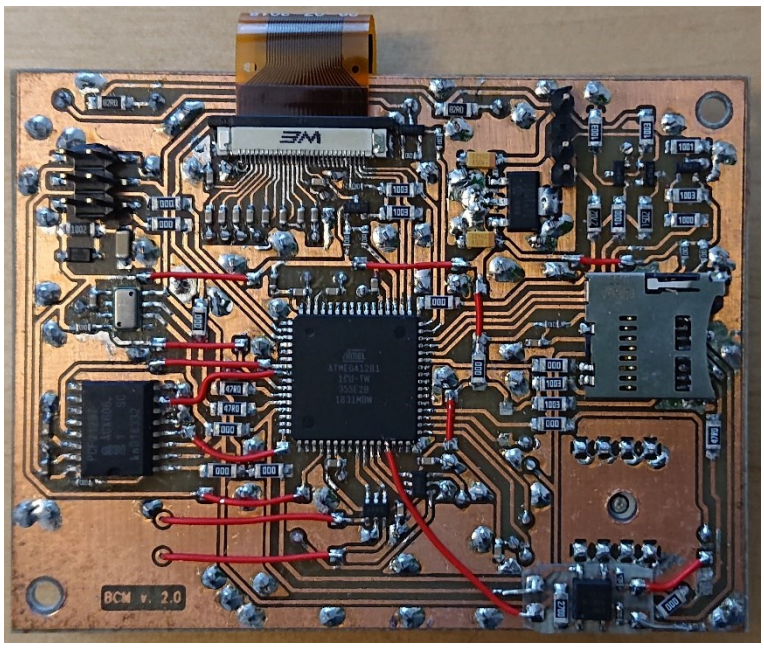
Levyn syövytys toteutettiin vedestä (H_2O), suolahaposta (HCl) ja vetyperoksidista (H_2O_2) muodostetulla liuoksella. Liuoksen valmistus aloitettiin lisäämällä 80 ml kylmää vettä. Veteen lisättiin 30 ml suolahappoa ja lopuksi 30 ml vetyperoksidia. Liuoksen voimakkuutta on mahdollista säätää veden ja muiden ainesosien suhdetta muuttamalla. Lisäksi aineiden lämpötila vaikuttaa syövytysreaktion nopeuteen. Käyttämällä kylmiä aineita reaktio ei ala voimakkaana ja syöpymistä on helpompi seurata. Liian voimakas tai kuuma seos syövyttää helposti liikaa kuparia, jolloin etenkin pienet yksityiskohdat saattavat kadota hyvinkin nopeasti. Syövytysprosessi aloitettiin upottamalla kehitetty piirilevy syövytysliuokseen. Prosessin seurantana toimi visuaalinen tarkistus. Syövytysprosessi ei mainittavasti kärsinyt, vaikka levy nostettiin liuoksesta välillä pois. Optimituloksen aikaikkuna on usein hyvin pieni ja varmuuden vuoksi lisää syövyttämällä käy usein niin, että syöpymistä tapahtuu liikaa.

Syövytyksen jälkeen piirilevyyn porattiin tarvittavat reiät ja jäljelle jäänyt fotoresisti poistettiin asetonilla. Tarvittaessa kuparia voi puhdistaa myös teräsvillalla. Lopuksi paljaille kuparipinnoille suihkutettiin hapettumista estävä ja juottamista helpottava pinnoite Contact Chemie SK10.

3.5 Erityishuomioita piirilevyn kokoonpanossa

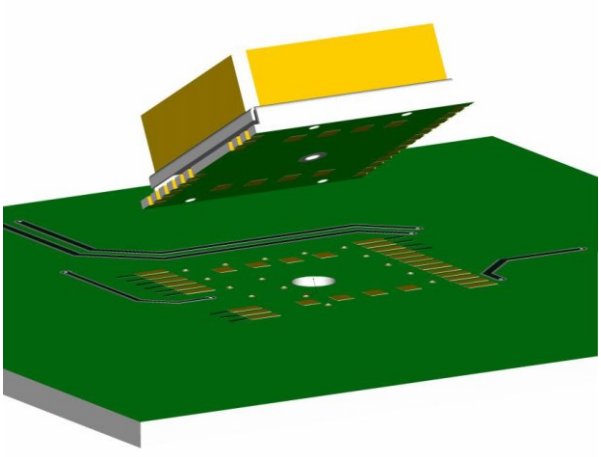
Piirilevyn kokoonpano suoritettiin kokonaan käsin juottamalla pienimpiä yksityiskohtia myöden. Kokoonpano vaati tarkkuutta ja vakaata kättä, mutta ei aiheuttanut varsinaisia ongelmia. Laitteessa oli kuitenkin kaksi komponenttia, joita ei ole tarkoitettu käsin juotettavaksi. Ilmanpaineanturissa oli liitospinnat ainoastaan komponentin alapinnalla, ja komponentti oli tarkoitettu asennettavaksi tinatahnan päälle, jolloin uunissa lämmitettäessä muodostuu kontakti komponentin ja piirilevyn välille. Tällaista vaihtoehtoa ei ollut prototyyppivaiheessa käytettävissä, joten piirilevylle juotettiin ensin tasainen tinakerros jokaisen padin kohdalle. Tämän jälkeen komponentti asetettiin ja pidettiin paikallaan. Komponentti saatiin tarttumaan ylimääräisen sivusta syötetyn tinan ja kuuman kolvin avulla. LCD-näyttö kiinnittyy piirilevylle FFC-kaapelin avulla. Kaapelia varten levylle juotettiin ZIF-liitin, jonka nastojen leveys on 0,3 mm ja etäisyys toisistaan 0,5 mm. Liitin juotettiin levylle varovasti pienen tinamäärän avulla. Juottamisen jälkeen liittimen nastat siivottiin ylimääräisestä tinasta tinaimunauhan avulla. Näin varmistettiin kontakti ilman oikosulkuja pienille yksityiskohdille. Liittimen päissä on suuret levylle

erikseen juotettavat kiinnityskohdat, joten vähäinen tinamäärä liittimen nastoissa ei vaikuta tukevuuteen. Valmiin piirilevyn yläpuoli koottuna on esitetty kuvassa 3.2.



Kuva 3.2 *Valmiin piirilevyn yläpuoli.*

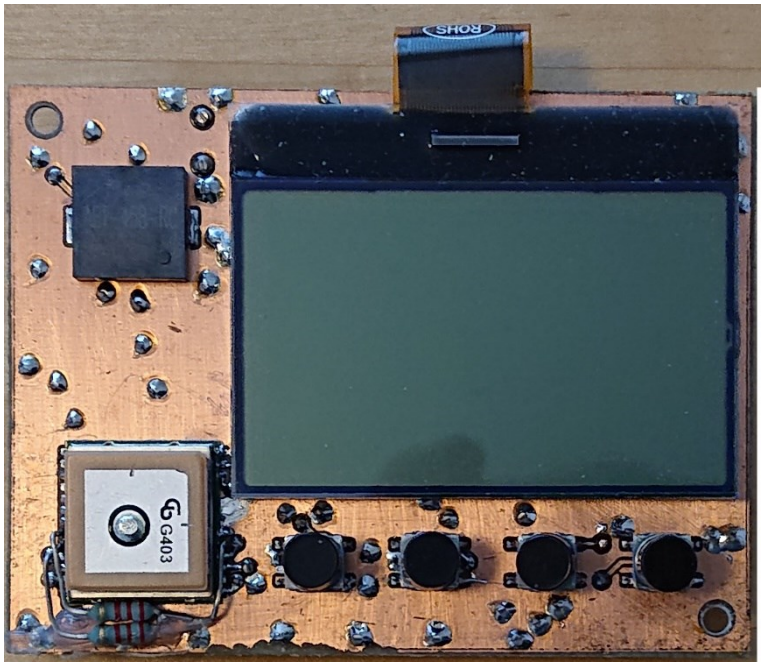
GPS -moduulissa on normaalien sivulla sijaitsevien liitoskohtien lisäksi pohjassa kahdeksan padia, jotka tulee liittää maatasoon. Padit ovat täysin piilossa, eikä edellä mainittu juottamiskeino ollut mahdollinen. Liitos tehtiin käyttämällä nestemäistä metallia, joka varmistaa sähköisen liitoksen, mutta ei tartu kiinni. Moduuli juotettiin reunoiltaan kiinni 22 kohdasta, joten pysyvyyden kannalta ongelmia ei tule. Piirilevyyn porattiin myös ylimäärinen 3 mm reikä GPS-moduulin alle RF-syöttöpisteen kohdalle (kuva 3.3). Tällä estetään häiriöiden kytkeytyminen maatasoon.



Kuva 3.3. *Piirilevyn rakenne GPS-moduulin kohdalla.[8]*

LCD-näyttö kiinnittyy piirilevylle FFC-kaapelin lisäksi neljällä piirilevyn läpi juotettavalla jalalla, joiden kautta kulkee sähkö taustavalolle. Näytön kiinnitystä vahvistettiin lisäksi

teippaamalla näyttö taustastaan kiinni piirilevyyn. Kuvassa 3.4 on esitetty piirilevyn alapuoli koottuna. Kuvassa näkyy hyvin useita maatasoon juotettuja läpivientejä, jotka on juotettu suoraan kupariin pelkän reiän avulla.



Kuva 3.4 *Valmiin piirilevyn alapuoli.*

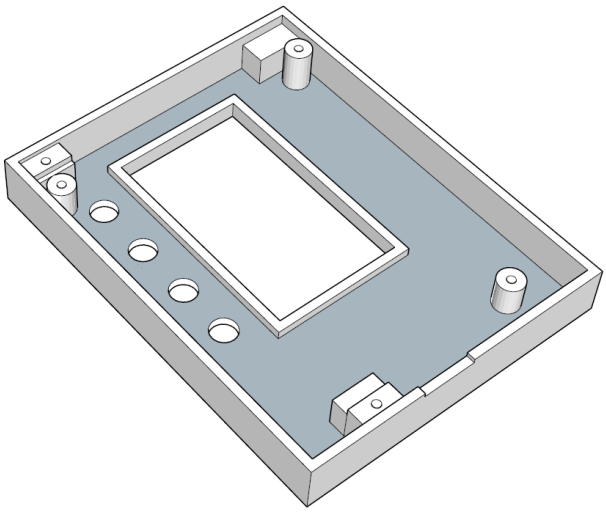
4. KOTELOINTI JA KIINNITYS

Testausta varten laite piti saada kiinnitettyä polkupyörään, joten prototyypille tarvittiin kotelo. Kotelo valmistettiin 3D-tulostamalla.

4.1 Kotelon suunnittelu

3D-tulostusta varten kotelo mallinnettiin ensin erillisellä ohjelmalla. Mallintamista varten tarjolla oli useita ilmaisia ohjelmia esimerkiksi OnShape ja Google SketchUp, joista jälkimmäisen käyttöliittymä vaikutti sopivammalta, joten kotelo suunniteltiin sillä.

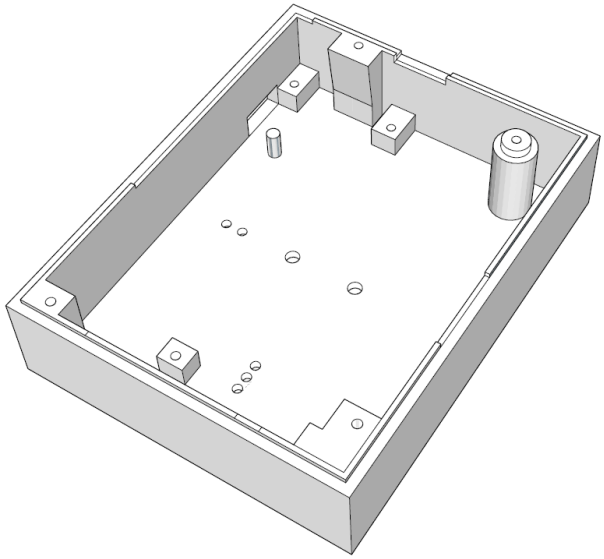
Kotelo koostuu kahdesta erillisestä kappaleesta, jotka kiinnitetään toisiinsa ruuveilla. Ruuvikiinnitystä varten kotelon ulkomitat kasvavat pienimmästä mahdollisesta 5 mm – 10 mm, mutta ruuvikiinnitys mahdollistaa helpon purkamisen ja kasaamisen testausvaiheessa. Kotelon kansiosaan (kuva 4.1) kiinnitetään piirilevy, jossa LCD-näyttö ja painonapit ovat kiinteästi asennettuina.



Kuva 4.1. *Kotelon kansiosa.*

Kotelon pohjakappaleeseen (kuva 4.2). kiinnitetään akku ohjausmoduuleineen. Takakappaleeseen on myös tehty reiät, joista ilma pääsee kulkemaan ilmanpaineanturille, ja toiset reiät, joista näkyy latausvaiheessa latauksesta kertova punainen valo ja valmiista latauksesta kertova vihreä valo. Pohjakappaleeseen

kiinnitetään myös kiinteästi muotokappale, jolla laite on mahdollista kiinnittää suoraan polkupyörän ohjauskannattimeen.



Kuva 4.2. *Kotelon pohjakappale.*

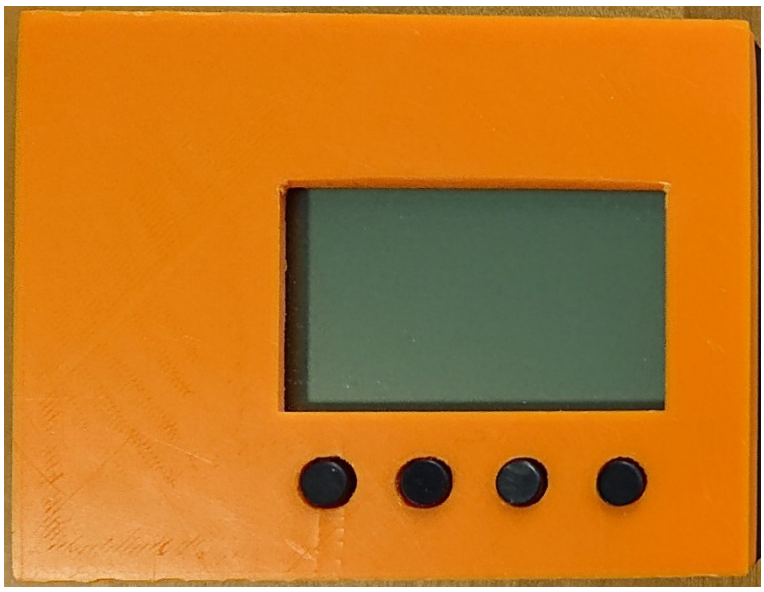
4.2 Kotelon valmistus

Kotelo valmistettiin Fablab Tampereen tiloissa Prusa I3 Mk3 -tulostimella käyttäen materiaalina PLA:ta. Tulostusta varten 3D-mallit viipaloitiin ohjelmallisesti. 3D-tulostin tulosti lopullisen kappaleen viipale kerrallaan. Viipaleen paksuutta säätämällä voidaan vaikuttaa lopullisen kappaleen pinnanlaatuun. Pienempi paksuus tuottaa tarkemmin valmistuneen kappaleen, mutta tulostukseen kuluu huomattavasti enemmän aikaa. Prototyypin kotelo tulostettiin normaaleilla laatuasetuksilla, koska ylimääräiselle tarkkuudelle ei ollut tarvetta. Viipalointi suoritettiin vapaan lähdekoodin Slic3r-ohjelmalla.

4.3 Kokoonpano

Laitteen kokoonpano 3D-tulostuksen jälkeen sujui pääosin ongelmitta. Kaikki reiät ja tukipisteet olivat kohdallaan ja ruuvikiinnitys onnistui suunnitellusti. Kaksi pientä aukkoa laitteen sivuilla tulostuivat epähuomiossa ulkopintaan asti ja lisäksi painonappien reikien kohdistus oli puutteellinen. Reiät olivat mallinnuksessa oikealla kohdalla ja oikean

kokoiset, mutta lopullisessa kappaleessa reiät olivat liian pienet ja kaksi niistä oli 0,5 mm sivussa (kuva 4.3).

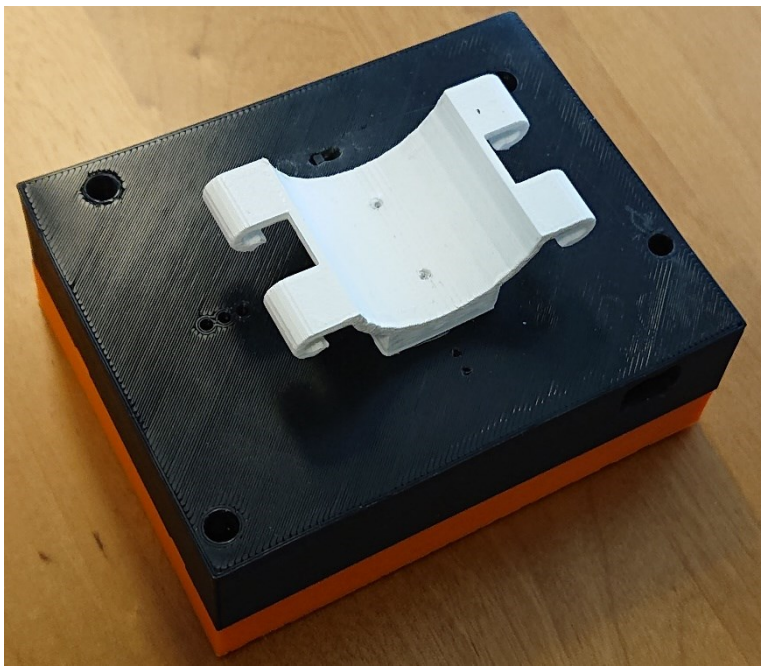


Kuva 4.3 *Valmis laite etupuolelta kuvattuna.*

Yksi selvä suunnitteluvirhekin oli jäänyt kotelon pohjakappaleeseen. Kappaleen kyljessä oleva latauspistokkeen aukko oli korkeussuunnassa väärällä puolella akkumoduulin piirilevyä, joten aukkoa piti suurentaa käsin.

Laitteen takakanteen lisättiin myös ylimääräinen katkaisin, jolla akkujännite pystyttiin irrottamaan laitteesta kokonaan koteloa avaamatta. Tästä oli lopulta hyötyä ohjelman testausvaiheessa tilanteissa, joissa ohjelmisto jäi jumiin. Takakanteen kiinnitettiin myös erillinen 3D-tulostettu osa, jolla laite oli helppo kiinnittää polkupyörän ohjauskannattimeen. Kuvassa 4.4 näkyy edellä mainittu valkoinen kannatin. Kannattimen yläpuolella on ylimääräinen virtakytkin ja alapuolella pienet reiät, joista

toisessa palaa latauksen aikana punainen valo ja toisessa vihreä valo, kun lataus on valmis. Kannattimen vasemmalla puolelle olevat reiät ovat paineanturin kohdalla.



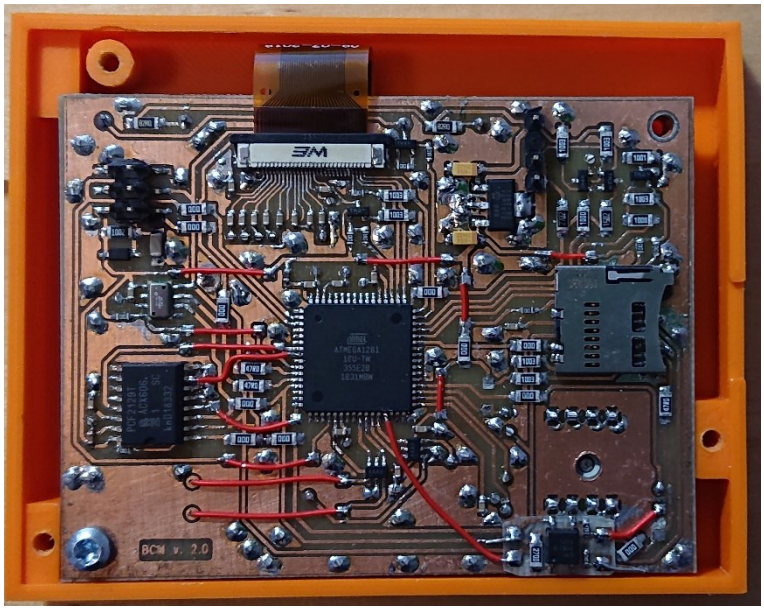
Kuva 4.4 *Valmis laite takaapäin kuvattuna.*

4.4 Erityishuomioita 3D-tulostetusta kotelosta

Testausten jälkeen laite purettiin osiin valokuvien ottamista varten. Ennen kuvien ottamista laite ehti kuitenkin olla purettuna noin yhden kuukauden. Tänä aikana kotelon osat olivat normaalissa huoneenlämmössä kirjoituspöydällä eikä niihin osunut esimerkiksi suoraa auringon valoa. Koteloa uudelleen koottaessa osissa havaittiin selvää muodon muutosta. Kuvassa 4.5 vasemmassa yläkulmassa näkyy, kuinka kotelon takaosan yksi kulma on painunut selvästi sisäänpäin. Vastaavasti kuvassa 4.6 oikeassa yläkulmassa näkyy samanlainen sisään painuminen. Kun laite on koottuna, ovat nämä kohdat vastakkain ja muodonmuutoksesta huolimatta laite on edelleen koottavissa. Testien aikana laite ei pudonnut tai saanut ulkoisia iskuja, mutta oli altistuneena suoralle

auringon paisteelle. Tämän perusteella 3D-tulostettavaan rakenteeseen pitäisi selvästi suunnitella sisäisiä tukirakenteita muodonmuutoksen ehkäisemiseksi.

Kuva 4.5 *Muotoa muuttanut takakansi.*



Kuva 4.6 *Muotoa muuttanut etukansi.*

5. OHJELMISTO

Ohjelmisto toteutettiin kokonaisuudessaan C-ohjelmointikielen avulla. Suurin osa ohjelmiston toiminnasta on tietoliikenteen ohjausta ja hallintaa eri komponenttien välillä sekä komponenteilta saadun tiedon tulkintaa. Ohjelmisto suorittaa myös laskentaa, jolla piireiltä saatua tietoa muutetaan sopivaan muotoon. Ohjelmisto toteutettiin modulaarisena, jolloin pääohjelma säilyi yksinkertaisena ja helppolukuisena. Toiminnot jaettiin erillisiin moduuleihin niin, että esimerkiksi LCD-näytön, GPS-moduulin ja SD-kortin osuudet ohjelmistosta erotettiin laajoina kokonaisuuksina omiin moduuleihinsa. Pienemmille toiminnoille tehtiin omat funktionsa ja funktiot kerättiin omaan selkään moduuliinsa.

5.1 Protokollat ja moduulit

5.1.1 NMEA 0183

NMEA 0183 on standardi, joka määrittelee elektronisen signaalin vaatimukset, tiedonsiirtoprotokollan ja ajoituksen sekä lauserakenteet vesikulkuneuvoissa käytettävälle 4800 baudin sarjaliikenteelle [10]. Tässä työssä GPS-moduuli lähettää tietoa tämän standardin mukaisesti ja koska käytössä on tietoliikenteen osalta SPI-väylä, tarvitaan standardia ainoastaan lauserakenteiden tulkintaan. Standardia ei käytetä signaalin tai johdotuksen määrittelyyn, koska kyseessä ei ole kahden NMEA-standardia käyttävän laitteen välinen dataliikenne.

NMEA-lauseessa käytetään standardin mukaisesti ASCII-merkkejä, jotka lähetetään siten, että eniten merkitsevä bitti on aina nolla. Sallittu merkistö koostuu kaikista tulostuskelpoisista ASCII-merkeistä, jotka sijoittuvat merkistössä välille HEX 20 – HEX 7E. Lisäksi käytössä on 10 varattua merkkiä. Tärkeimmät ja tässä työssä käytetyt varatut merkit ja niiden merkitykset on esitetty taulukossa 5.1. Muita varattuja merkkejä ovat !, \, ^, ~ ja <del>. Näistä merkeistä vain ^ ja ! ovat käytössä ja niitäkin tarvitaan hyvin harvoin. Muita kuin edellä mainittuja sallittuja ja varattuja merkkejä ei saa standardin mukaan käyttää.

Taulukko 5.1 *NMEA-lauseen tärkeimmät varatut merkit.*

Merkki	Heksa	Desimaali	Merkitys
<CR>	0D	13	Carriage return, lauseen loppumerkki
<LF>	0A	10	Line feed, rivinvaihto
$	24	36	Parametrilauseen alkumerkki
*	2A	42	Tarkistussumman alkumerkki
,	2C	44	Tietokentän erotin

Maksimimerkkimäärä NMEA-lauseessa on 82 siten, että aloittavan merkin $ ja lopettavien merkkien <CR><LF> välissä on enintään 79 merkkiä. Lauseessa on oltava vähintään yksi tietokenttä, joka määrittelee tiedon lähettäjän ja lauseen tyypin. Kenttien maksimimäärää rajoittaa ainoastaan merkkien maksimimäärä.

Lauseen alussa tulee aina olla osoitekenttä, joka alkaa joko merkillä $ tai !. Merkki $ aloittaa normaalin parametrejä sisältävän lauseen. Merkki ! aloittaa erityisen kapseloidun lauseen, jota tarvitaan jos lähetysnopeutta tarvitsee kasvattaa yli 38400 baudin. GPS-moduulin tapauksessa kapseloituja lauseita ei käytetä. Osoitekentän kaksi seuraavaa merkkiä ovat kirjaimia, jotka määrittävät minkä tyyppinen laite on lauseen lähettäjänä. GPS-moduulin tapauksessa nämä kirjaimet ovat GP. Seuraavat kolme merkkiä määrittelevät lauseen tyypin. Tässä työssä käytettävä GPS-moduuli lähettää seitsemän erilaista lausetta, jotka on esitelty taulukossa 5.2.

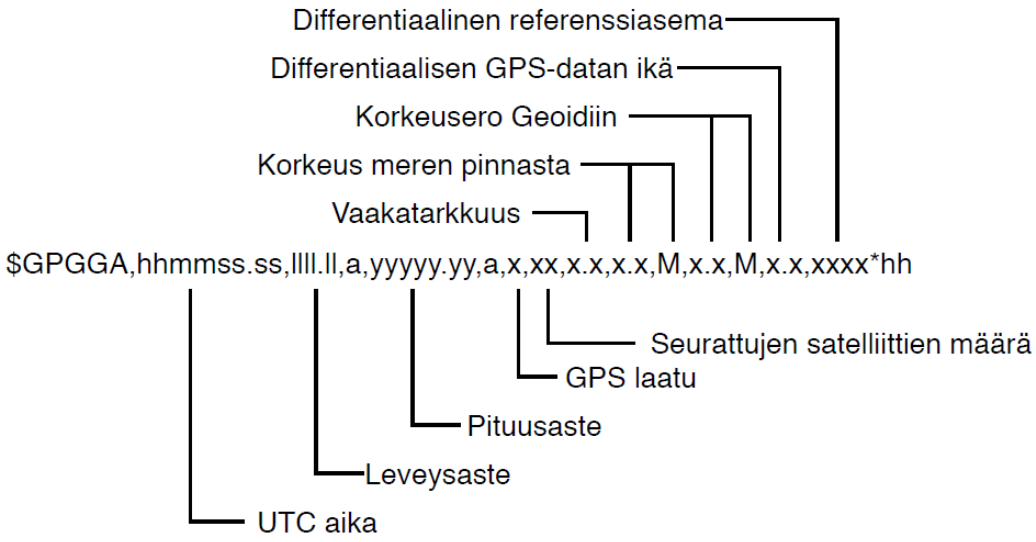
Taulukko 5.2 *GPS-moduulin lähettämät NMEA-lausetyypit.*

Lause	Selitys
GGA	Global Positioning System Fix Data, tiedot GPS-yhteyden laadusta
RMC	Recommended Minimum Specific GNSS Data, suositeltu minimidata
GSA	GNSS DOP nad Active Satellites, lisätietoja paikannuksen tarkkuudesta
GSV	GNSS Satellites in View, tietoja näkyvillä olevista satelliiteista
GLL	Geographic Position - Latitude/Longitude, sijaintitiedot ja aika
VTG	Course over Ground and Ground Speed, suunta ja nopeus
ZDA	Time & Date, aika ja päivämäärä

Moduulin lähettämistä lauseista vain GGA ja RMC ovat käytössä tässä työssä. Nämä kaksi lausetta ovat rakenteeltaan useita tietoja yhteen kokoavia ja tarjoavat tarvittavat minimitiedot. Muista tarjolla olevista lauseista olisi saatavilla yksityiskohtaisempaa tietoa

esimerkiksi näkyvillä olevien satelliittien asemasta tai paikannuksen tarkkuudesta. Näitä ei kuitenkaan tarvita yksinkertaisessa nopeuden tarkkailussa ja matkan laskennassa.

GGA-lauseen rakenne on esitetty kuvassa 5.1. Pituus- ja leveysasteet koostuvat asteluvusta ja sitä seuraavasta merkistä *a*. Tämä merkki kertoo millä pallonpuoliskolla sijainti on ja voi leveysasteissa olla N tai S ja pituusasteissa E tai W.



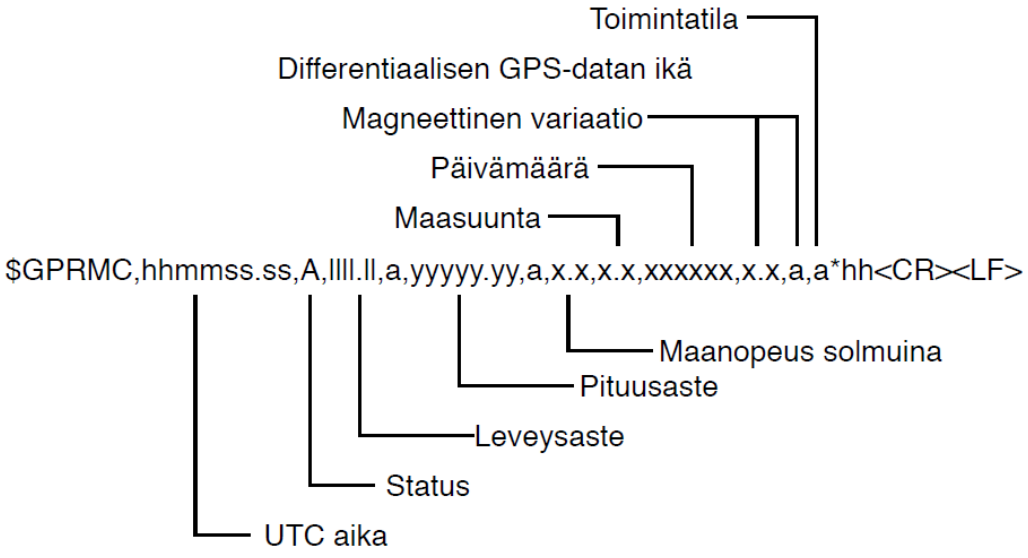
Kuva 5.1 *GGA-lauseen rakenne.*

GPS-yhteyden laadusta kertovat numerot ja niiden selitykset on esitetty taulukossa 5.3. Välillä 0 – 5 yhteyden laatu paranee numeron kasvaessa. Normaali ja differentiaalinen GPS-yhteys ovat molemmat normaalin tarkkuuden toimintatiloja. Nämä toimintatilat ovat maksuttomia ja kaikkien käytettävissä maailmanlaajuisesti. Precise Positioning Service on erittäin tarkka, mutta sen käyttö on suojattua ja rajattua sekä vaatii valtuudet. Tämä yhteys on esimerkiksi viranomaisten käytössä. Real Time Kinematic – yhteys on myös erittäin tarkka ja sitä käytetään esimerkiksi maanmittauksessa. Tässä työssä ohjelmisto tarkastelee yhteyden laatua vain tarkastamalla, onko laatu 0, jolloin yhteyttä ei ole, vai onko laatu nollaa suurempi, jolloin yhteys on olemassa. Laadun lisäksi GGA-lauseen tiedoista käsitellään aika, seurattujen satelliittien määrä ja korkeus.

Taulukko 5.3 *GPS-yhteyden laatumerkinnät GGA-lauseessa.*

Arvo	Selitys
0	Ei GPS-yhteyttä
1	Normaali GPS-yhteys
2	Differentiaalinen GPS-yhteys
3	Precise Positioning Service -yhteys
4	Real Time Kinematic -yhteys kokonaisluvuilla
5	Real Time Kinematic -yhteys liukuluvuilla
6	Arviotila
7	Manuaalisyöttötila
8	Simulaattoritila

RMC-lauseen rakenne on esitetty kuvassa 5.2. Lause sisältää osin samoja tietoja kuin GGA, mutta tärkeimpänä erona on maanopeus, joka saadaan lauseesta solmuina. Nopeus kilometreinä tunnissa saadaan kertomalla nopeus luvulla 1,852. Tässä työssä RMC-lauseen tiedoista käsitellään aika, leveysaste, pituusaste, maanopeus ja päivämäärä.

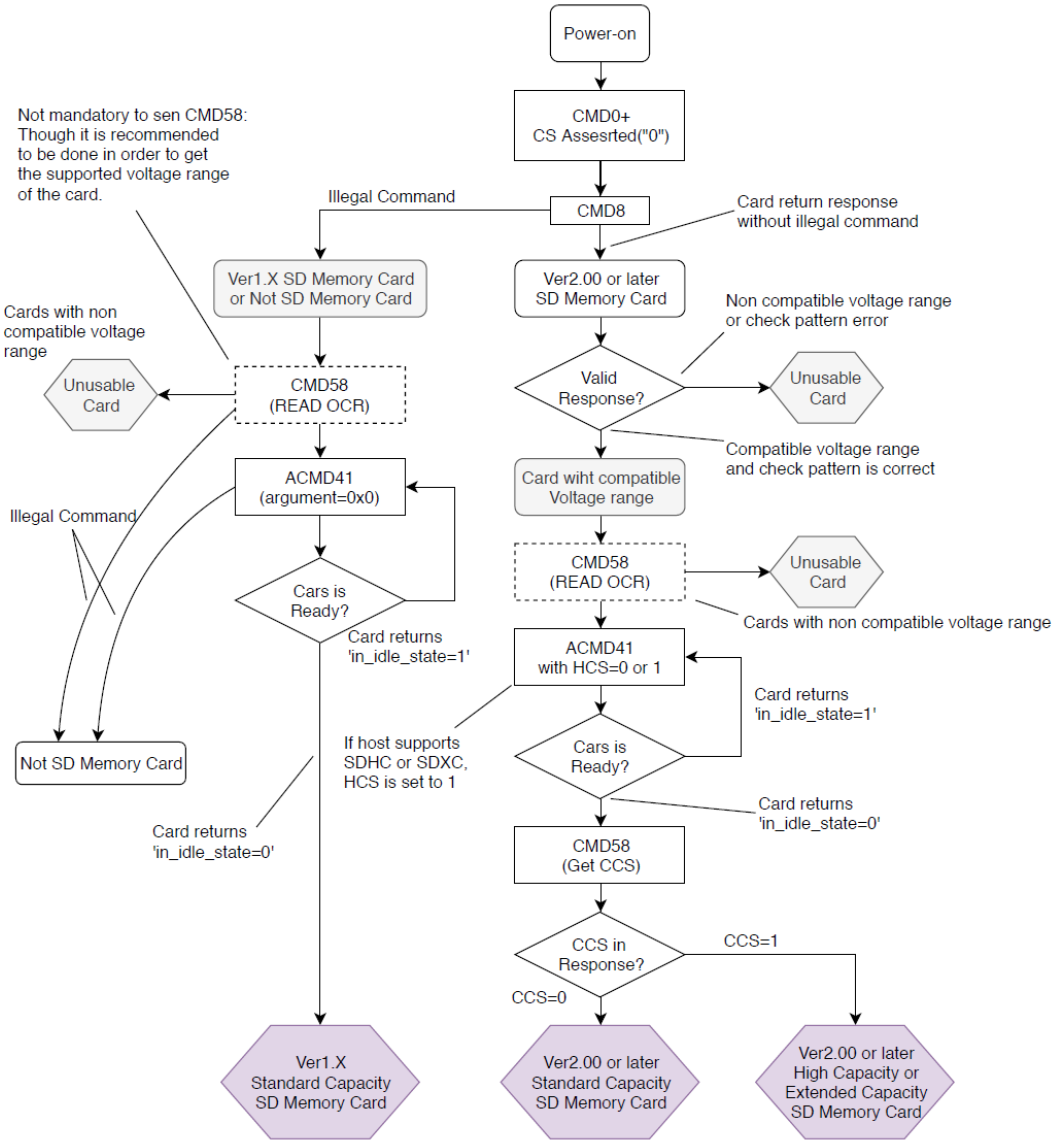


Kuva 5.2 *RMC-lauseen rakenne.*

5.1.2 SD-kortin alustus ja tiedostojärjestelmä

SD-kortti toimii virtojen kytkemisen jälkeen oletusarvoisesti SD-moodissa. SD-kortit tukevat kuitenkin myös SPI-muotoista tiedonsiirtoa. Tila valitaan kortin alustuksen yhteydessä asettamalla CS aktiiviseksi reset-komennon (CMD0) aikana. Kortti kuittaa

siirtymisen SPI-tilaan lähettämällä R1-vastauksen. Kuvassa 5.3 on esitetty SPI-moodin alustuksen kulku. Komennolla CMD8 varmistetaan, että SD-kortti on toimintakunnossa. Mikäli kortti ei tunnista komentoa on kyseessä vanhempi kortti, jolloin alustusprosessi on suppeampi. Jos kortti vastaa komentoon on kyseessä version 2.00 tai uudempi kortti. Näillä korteilla vastaus sisältää tiedon sopivasta käyttöjännitealueesta. Kaikkien korttien tapauksessa sopiva käyttöjännite varmistetaan vielä komennolla CMD58. Tämän jälkeen kortille lähetetään käsky ACMD41, joka käynnistää kortin alustuksen ja jonka vastauksen perusteella voidaan päätellä koska alustus on valmis. Kun vastaus on *in_idle_state=0,* on kortti käyttökunnossa. Uudemmille korteille lähetetään lopuksi komento CMD uudestaan CCS-tiedon (Card Capacity Status) lukemiseksi. Vastaus 0 kertoo tavallisesta ja vastaus 1 korkean kapasiteetin SD-kortista. [15]



Kuva 5.3 *SD-kortin alustuksen vuokaavio. [15]*

SD-kortin alustus on monivaiheinen prosessi, jossa lähetetään käskyjä, odotetaan vastauksia ja toimitaan niiden mukaisesti. Alustus on täysin standardin mukainen ja toteutettavissa ilman ulkopuolisia moduuleita. Kortin tiedostojärjestelmä on kuitenkin niin monimutkainen ja laaja kokonaisuus, että valmiin ulkopuolisen moduulin käyttö on täysin perusteltua. Yksi tällainen moduuli on vapaan lähdekoodin FatFs. Moduuli sisältää myös SD-kortin alustusfunktiot, joten niitä ei kirjoitettu erikseen itse.

SD-korttia olisi mahdollista käyttää tavallisen muistin tapaan niin, että kortin tiettyyn osoitteeseen kirjoitetaan haluttua dataa ja myöhemmin data luettaisiin samasta osoitteesta. SD-kortin on kuitenkin tarkoitus toimia siirrettävänä mediana laitteen ja tietokoneen välillä, joten kortille on luotava tiedostojärjestelmä, jota myös tietokone osaa lukea. Windows-pohjaisissa PC-tietokoneissa pitkään käytössä ollut FAT-tiedostojärjestelmä on laajasti tuettu ja hyvin usein uudet SD-kortit käyttävät tehtaan jäljiltä tämä tiedostojärjestelmää.

Alkuperäinen FAT on Microsoftin vuonna 1977 julkaisema tiedostojärjestelmä, joka tuki enintään 32 megatavun tiedostojärjestelmiä. Parannettu versio FAT16 esiteltiin vuonna 1984 ja tämä versio tuki 2 gigatavun kokoisia tiedostojärjestelmiä. Edelleen parannettu FAT32 esiteltiin vuonna 1996. FAT32 tukee enintään 8 teratavun kokoisia tiedostojärjestelmiä. FAT32 on riittävä tukemaan kaikkia tällä hetkellä saatavilla olevia SD-kortteja, joten sen käyttö on perusteltua. Tiedostojärjestelmän nimessä esiintyvät numerot kuvaavat suoraan FAT-rakenteen kokoa levyllä bitteinä. [19]

FAT-tiedostojärjestelmä rakentuu neljästä eri osiosta [3][9]:

- Käynnistyssektori (*engl. Boot Sector*) sijaitsee levyn alussa ja sisältää järjestelmän tärkeimmän tietorakenteen BPB (BIOS Parameter Block), johon on tallennettu tiedostojärjestelmän parametrit.
- FAT-alue, joka määrittelee miten sektoriryppäät (*engl. cluster*) on järjestetty. Tämä tieto on kahdennettu, jotta yksittäinen virhe ei sekoita koko järjestelmää.
- Juurihakemistoalue (*engl. Root Directory Region*) sisältää informaation tiedostoista ja hakemistoista. FAT32-järjestelmässä tämä alue on yhdistetty tietoalueeseen.
- Tietoalue (*engl. Data Region*) sisältää levylle tallennetut tiedostot. Suurin osa levystä on tätä aluetta.

FatFs-moduuli hoitaa kaikki tiedostojärjestelmään liittyvät toimenpiteet ja sisältää laajasti ominaisuuksia tietojärjestelmän tehokkaaseen käyttöön. Tässä työssä SD-kortille tallennetaan yksinkertaisia ja pienikokoisia tekstitiedostoja, joten suurin osa moduulin

potentiaalista jää käyttämättä. Alustuksen lisäksi käyttöön otettiin kolme funktiota *f_*open¸*f_*write ja *f_close.* Nimiensä mukaisesti *f_open* ja *f_close* avaavat ja sulkevat tiedoston SD-kortilla. Funktio *f_write* kirjoittaa tietoa tiedostoon. Liitteessä 7 on kuvattuna kokonainen funktio, jolla laite avaa tekstitiedoston SD-kortille, kirjoittaa tarvittavat tiedot ja lopulta sulkee tiedoston. Tiedoston nimi muodostetaan päivämäärästä ja juoksevasta numerosta.

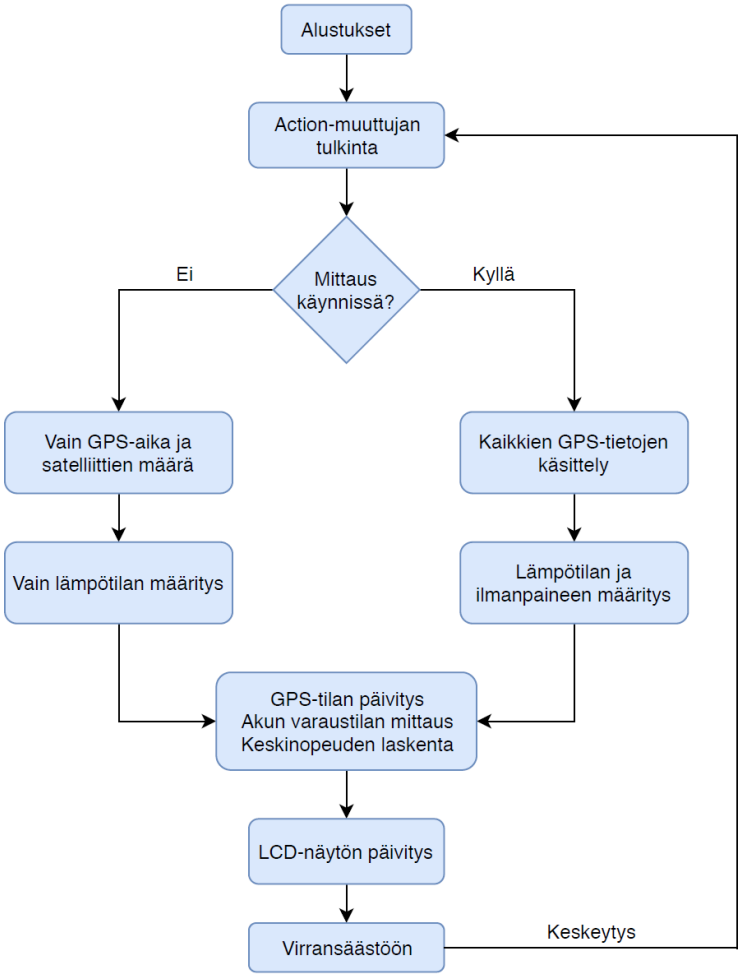
Vaikka FatFs-moduuli onkin laajuutensa takia suurikokoinen, ei sen mikrokontrolleriin mahtumisen kanssa kohdattu ongelmia. Moduuli sisältää noin 10000 riviä ohjelmakoodia, joten on selvää, että rajoitettujenkin toimintojen uudelleen kirjoittaminen itse olisi vaatinut kohtuuttoman paljon aikaa.

5.2 Rakenne

Ohjelmiston pääohjelmana toimii ikuinen silmukka, jota toistetaan jatkuvasti laitteen ollessa päällä. Pääohjelma suorittaa yhden kierroksen ja siirtyy syvään virransäästötilaan odottamaan keskeytystä, joka aloittaa ohjelman toiminnan uudestaan. Kuvassa 5.4 nähdään pääohjelman toiminnallinen rakenne.

Ohjelman suoritus alkaa sulautetuille järjestelmille ominaisella järjestelmän alustuksella. Alustus suoritetaan akkuvirran kytkemisen jälkeen vain kerran. Alustus sisältää ohjelman muuttujien määritykset ja mikrokontrollerin porttien, SPI-väylän, AD-muuntimen, keskeytysten ja ajastimien alustuksen. Ilmanpaineanturin käyttöönottoa varten piirin sisäisistä rekistereistä luetaan tehdaskalibrointiarvot, joita käytetään myöhemmin sekä lämpötilan että ilmanpaineen laskennassa. LCD-näytön alustus vaatii monivaiheisen ja oikeassa järjestyksessä suoritettavan komentosarjan SPI-väylän kautta. Näytön sisäinen boost-hakkuri kytketään päälle ja sisäinen bias asetetaan arvoon 1/7. Lisäksi näytön ohjelmallinen kontrastin säätö asetetaan sopivaan puolivälin arvoon, näytön pysty- ja vaakasuunnat valitaan ja kursori siirretään vasempaan yläkulmaan. GPS-moduulin alustus on yksinkertainen ja koostuu vain resetoinnista ja 150 ms -mittaisen käynnistyspulssin antamisesta. Ensimmäisen käynnistyksen yhteydessä resetin jälkeen moduuli valitsee dataliikenteen toimintatavan GPIO6- ja GPIO7-porttien tilojen perusteella. SPI-väylää varten varmistetaan ohjelmallisesti, että GPIO6 on tilassa *low* ja GPIO7 tilassa *high*. Ennen seuraavaa resettiä dataliikenteen valinta ei vaihdu, vaikka moduuli sammutetaan välillä. Myös kalenteripiiri alustetaan SPI-väylän kautta annettavilla komennoilla. Piirille suoritetaan OTP-alustus, jossa piiri lukee sisäisestä EPROM-muistista tehdaskalibrointiarvot. Tämän jälkeen piiriltä otetaan käyttöön 1 Hz -kellosignaali ja keskeytyssignaali sekä piiri määritetään käyttämään jatkuvaa käyttöjännitettä ilman akkuvarmennusta. Piirissä on sisäänrakennettu ominaisuus, jolla

se osaisi automaattisesti siirtyä käyttämään ulkoista akkua, jos käyttöjännite katkeaa, mutta tälle ominaisuudelle ei ole tarvetta tässä työssä. Aikaa ja päivämäärää ei ole tarpeen saada säilymään, jos akku syystä tai toisesta tyhjenisi kokonaan.



Kuva 5.4 *Pääohjelman vuokaavio.*

Alustuksen jälkeen ohjelma suorittaa yksinkertaisen *action*-muuttujan tulkinnan. Tämän muuttujan tilaa muutetaan painikkeiden avulla ja sen kautta suoritetaan toimenpiteitä, jotka eivät ole käyttöliittymän kannalta aikakriittisiä. Tämän muuttujan kautta ohjattavia toimenpiteitä ovat keskinopeuden laskenta stop-tilanteessa, laitteen sammutus, SD-kortille tallennus ja tietojen nollaus.

Laitteen virrankulutuksen hallintaa varten seuraavan vaiheen toiminta riippuu siitä, onko mittaus käynnissä vai ei. Jos mittaus ei ole käynnissä, haetaan GPS-moduulilta ja ilmanpaineanturilta vain vähimmäismäärä tietoa, jolloin prosessoriaikaa ja sen kautta virtaa ei käytetä turhaan laskentaan. Haettavia tietoja ovat aika, satelliittiyhteyden tila ja

laatu sekä lämpötila. Jos mittaus on käynnissä, haetaan molemmilta piireiltä kaikki tarvittavat tiedot ja suoritetaan täydellinen laskenta.

Seuraavassa vaiheessa päivitetään tieto GPS-yhteyden tilasta, mitataan akun varaustila ja lasketaan keskinopeus. Varaustilan mittaus ja keskinopeuden laskenta suoritetaan ajastimen ja *update_data* -muuttujan avulla 30 sekunnin välein.

Lopulta kaikki edellä määritetty tieto päivitetään LCD-näytölle ja laite siirtyy syvään virransäästötilaan odottamaan keskeytystä, jonka perusteella ohjelman suoritus alkaa alusta. Pääasiallinen keskeytys tähän tarkoitukseen on kalenteripiirin 1 Hz -kellosignaali. Normaalitilassa laite päivittää tiedot yhden sekunnin välein. Päivitysväli on valittu GPS-moduulin perusteella, koska sen tehdasarvo päivitystaajuudelle on 1 Hz. Kellosignaalin lisäksi myös kalenteripiirin keskeytyssignaali tai nappien painaminen herättää laitteen.

5.3 Pääfunktiot

5.3.1 GPS-moduuli

GPS-moduulin tietojen käsittely suoritetaan neljässä eri vaiheessa. NMEA-lauseet haetaan moduulista SPI-väylän avulla, tulkitaan, muunnetaan ja lopulta niiden perusteella lasketaan kuljettu matka. Tietojen hakuvaiheessa tietoa siirretään niin kauan, kunnes sekä GGA- että RMC-lauseet on haettu tai moduulin FIFO-rekisteri on tyhjä. NMEA-lauseet ovat 82 merkin pituisia ja jokaisen lauseen lopussa on lauseen lopusta kertova merkki ja rivinvaihto. Rivinvaihdon perusteella lauseet tallennetaan mikrokontrollerin muistiin rivi kerrallaan. Ohjelma tulkitsee rivin loppuneeksi myös, jos 82 merkkiä on jo luettu, vaikka rivinvaihtomerkki puuttuisi. Näin mahdollisessa häiriötilanteessa ei tapahdu ylivuotoja.

Tietojen lukemisen jälkeen suoritetaan lauseiden tulkinta. Vain GGA- ja RMC-lauseet tulkitaan, koska ne sisältävät kaiken tarvittavan tiedon laitteen toimintaa varten. Jokainen NMEA-lause alkaa lausetta kuvaavalla termillä *$GPGGA*, jossa kolme viimeistä merkkiä kertovat mikä lause on kyseessä. Tämä jälkeen lause sisältää dataa pilkuilla eroteltuna. Lauseiden tulkinta aloitetaan tarkastamalla sisältääkö lause dataa pilkkujen välissä. Jos lause alkaa heti pelkkinä pilkkuina, lopetetaan tulkinta, koska lause on tyhjä. Muussa tapauksessa koko lause käydään läpi merkki kerrallaan ja tarvittavat osat pilkkujen välisestä datasta tallennetaan muuttujiin. Tulkinta perustuu lauseiden vakiorakenteeseen, jossa haluttu data on aina samassa kohdassa lausetta. Esimerkkinä

seuraava *while*-silmukka lukee RMC-lauseesta leveysasteen, joka on lauseessa muodossa ddmm.mmmm:

```
while (data[i] != ',' && i < 82) {
        nmea_tmp[tmp_index] = data[i];
        ++i;
        ++tmp_index;
}
```

Silmukassa muuttuja *i* on indeksi, joka saa arvoja väliltä 0 – 81. Indeksi kuvaa missä kohdassa lausetta tulkinta kulkee. Muuttuja *data* sisältää GPS-moduulilta luetun lauseen. Indeksin avulla lauseesta tallennetaan haluttu kohta väliaikaismuuttujaan *nmea_tmp*. Silmukkaa suoritetaan, kunnes tulkittava merkki on pilkku, tai lause on lopussa.

Tulkinnan jälkeen datalle suoritetaan tarvittavat muunnokset. Tässä vaiheessa kaikki saatu tieto on ASCII-merkkeinä, jolloin muuttujan tyyppi on *char*. *Char* voidaan helposti muuttaa kokonaisluvuksi vähentämällä siitä merkki *'0'*. Tämän jälkeen numeroilla voidaan suorittaa normaaleja laskutoimituksia. Esimerkiksi aika saadaan RMC-lauseesta muodossa *hhmmss*. Kokonaisluvuksi muuttamisen jälkeen tunnit saadaan laskettua $h_1 * 10 + h_2$. Matkan laskentaa varten paikkakoordinaateille suoritetaan muunnos radiaaneiksi. Paikkakoordinaatit saadaan lauseesta muodossa *ddmm.mmmmX*, jossa d-kirjaimella merkityt osuudet ovat suoraan asteita ja m-kirjaimella merkityt asteminuutteja. Lopussa olevalla X-kirjaimella merkitään pallonpuolisko. Pallonpuolisko tunnistetaan merkeillä N, S, E ja W. Ennen radiaaneiksi muuttamista paikkakoordinaatti täyttyy muuttaa kokonaisuudessaan asteiksi. 60 asteminuuttia vastaa yhtä astetta, joten koordinaatti voidaan muuttaa asteiksi kaavalla:

$$dd + \frac{mm.mmmm}{60} \tag{7}$$

180° vastaa π radiaania, joten asteet saadaan muunnettua radiaaneiksi kertomalla asteet luvulla $\pi/180$. Tässä vaiheessa laskentaa lukuarvot skaalataan ylöspäin kertomalla ne luvulla 10^6. Skaalauksen avulla laskutoimitukset saadaan jatkossa

suoritettua kokonaisluvuilla. AVR-arkkitehtuurissa kokonaisluvuilla laskeminen on nopeampaa ja liukulukujen tarkkuus on hyvin rajattua.

Muunnoksen lopuksi huomioidaan pallonpuolisko siten, että arvot S ja W aiheuttavat lopullisen tuloksen kertomisen luvulla -1.

Lopuksi lasketaan kahden viimeisimmän koordinaatin ja kappaleessa 2.1 kuvatun kaavan avulla pisteiden välinen etäisyys.

Laskennan jälkeen viimeisimmät leveys- ja pituuskoordinaatit tallennetaan muuttujiin *ex_lat* ja *ex_lon* seuraavaa laskentakierrosta varten.

5.3.2 Ilmanpaineanturi

Ilmanpaineanturi mittaa paineen lisäksi lämpötilaa ja molemmat arvot saadaan digitaalisessa muodossa SPI-väylän kautta. Mittaus aloitetaan antamalla väylän kautta käsky AD-muunnoksen aloituksesta. Muunnos suoritetaan erikseen paineelle ja lämpötilalle. Käskyn yhteydessä määritellään käytettävä muunnoksen resoluutio. Käytettävä resoluutio vaikuttaa mittaustulosten tarkkuuteen ja muunnoksen vaatimaan aikaan. Taulukossa 5.4 on esitetty valmistajan antamat tiedot resoluution vaikutuksesta. Taulukosta voidaan havaita, että pienimmälläkin resoluutiolla lämpötilan ja ilmanpaineen tarkkuus on varsin hyvä ottaen huomioon, että lämpötila esitetään mittarissa yhden desimaalin tarkkuudella. Ilmanpaineen pienin tarkkuus on 0,13 mbar, joka vastaa noin metrin korkeuseroa. Nousu- ja laskumetrien kertymistä mitataan metrin tarkkuudella, joten suuremman resoluution käyttö on perusteltua.

Alun perin laskennassa käytettiin täyttä resoluutiota, jolla saavutettiin 0,024 mbar tarkkuus. Kappaleessa 6.2 käsitellyssä ohjelmiston testauksessa havaittiin kuitenkin, että mittaustulos vaihtelee enemmän kuin resoluutio antaa olettaa, joten lopulta päädyttiin käyttämään resoluutiona arvoa 1024, jolla saavutetaan 0,054 mbar, eli noin 40 cm tarkkuus. Lämpötilan arvoa käytetään ilmanpaineen laskennassa, joten sen muunnokselle valittiin sama resoluutio 1024.

Taulukko 5.4 *Ilmanpaineanturin AD-muunnoksen resoluution vaikutus mittaustarkkuuteen. [17]*

OSR	Muunnoksen kesto [ms]	Lämpötilan resoluutio [°C]	Ilmanpaineen resoluutio [mbar]
256	0,54	0,012	0,13
512	1,06	0,008	0,084
1024	2,08	0,005	0,054
2048	4,13	0,003	0,036
4096	8,22	0,002	0,024

Ilmanpaineanturi ei anna SPI-väylän kautta arvoja, joita olisi mahdollista käyttää suoraan. Sekä ilmanpaineen että lämpötilan arvot ovat 24-bittisiä lukuja, jotka muutetaan laskennallisesti ilmanpaineeksi millibaareina ja lämpötilaksi celsiusasteina. Laskenta perustuu mitattuihin arvoihin ja piiriin tallennettuihin tehdaskalibrointiarvoihin. Ohjelmakoodissa laskenta on suoritettu C-kielen käskyillä, mutta koska laskennassa ohjelmoinnin osuus on tässä tapauksessa triviaalinen, esitetään kaavat alkuperäisessä muodossaan helpomman luettavuuden takia.

Lopullisen ilmanpainearvon laskenta koostuu viidestä vaiheesta. Kahdessa ensimmäisessä vaiheessa lasketaan lämpötilan arvo, jota käytetään kolmessa viimeisessä vaiheessa apuna. Lämpötilan laskennassa tarvittavien muuttujien tarkkuus on 32 bittiä, mutta ilmanpaineen laskennan aikana luvut kasvavat niin suuriksi, että tarvitaan 64 bitin tarkkuutta. Ilmanpaineen laskenta on 8-bittiselle mikrokontrollerille raskasta laskentaa, joten paineen arvo määritetään ohjelmassa vain silloin kun sitä oikeasti tarvitaan, kuten kappaleessa 5.2 on mainittu. Laskennassa tarvittavat kertoimet on esitetty taulukossa 5.5.

Taulukko 5.5 *Ilmanpaineanturin korjauskertoimet.*

Muuttuja	Kuvaus
C1	Paineen herkkyys
C2	Paineen poikkeama
C3	Paineen herkkyyden lämpötilakerroin
C4	Paineen poikkeaman lämpötilakerroin
C5	Referenssilämpötila
C6	Referenssilämpötilan lämpötilakerroin
D1	Digitaalinen paineen arvo
D2	Digitaalinen lämpötilan arvo

Lämpötilan laskennassa ensimmäisenä lasketaan lämpötilan muutos suhteessa referenssilämpötilaan:

$$dT = D2 - C5 * 2^8 \quad (8)$$

Lopullinen lämpötila saadaan kokonaislukuarvona siten, että luku 2000 vastaa lämpötilaa 20,00 °C. Lämpötila lasketaan kaavalla:

$$TEMP = 2000 + \frac{dT*C6}{2^{23}} \quad (9)$$

Ilmanpaineen laskennassa ensin lasketaan todellinen poikkeama mitatussa lämpötilassa kaavalla:

$$OFF = C2 * 2^{17} + \frac{C4*dT}{2^6} \quad (10)$$

Seuraavaksi lasketaan todellinen herkkyys mitatussa lämpötilassa kaavalla:

$$SENS = C1 * 2^{16} + \frac{C3*dT}{2^7} \quad (11)$$

Lopullinen lämpötilakompensoitu ilmanpainearvo saadaan kokonaislukuarvona siten, että luku 110002 vastaa ilmanpainetta 1100,02 mbar. Ilmapaine saadaan kaavalla:

$$P = \frac{D1*SENS}{2^{21}} - \frac{OFF}{2^{15}} \quad (12)$$

Lopuksi kappaleessa 2.2 esitetyllä kaavalla lasketaan korkeus nykyhetkellä ja edellisellä mittauskerralla ja näiden erotuksen perusteella päätellään, onko korkeussuunnassa liikuttu ylös- vai alaspäin. Tämän perusteella kerrytetään nousu- ja laskumetrejä kumulatiivisesti. Laskenta ja tiedon keräys suoritetaan senttimetrin tarkkuudella, mutta esitetään lopulta näytöllä yhden metrin tarkkuudella niin, että numero vaihtuu, kun kokonainen metri on tullut täyteen.

5.3.3 LCD-näyttö

LCD-näytön näkyvä osuus koostuu 128x64 pikselin suuruisesta alueesta. Jokaista pikseliä ohjataan päälle/pois-periaatteella yksitellen. Näytössä ei ole sisäänrakennettuna esimerkiksi fontteja. LCD-näytön ST7565P-ohjainpiiri on hyvin yleinen ja näytön ohjausta varten on saatavilla useita erilaisia valmiita kirjastoja. Mikään kirjasto ei olisi toiminut suoraan rakennettavassa laitteessa, vaikka vapaana lähdekoodina niiden käyttö olisi ollut mahdollista. Lisäksi valmiin kirjaston käyttäminen aiheuttaa helposti tilanteen, jossa ohjelmoija ei oikeasti ymmärrä miten kirjasto ja näytön ohjaus toimii, jolloin laitekohtaisten erikoisominaisuuksien rakentaminen vaikeutuu huomattavasti. Tästä

syystä LCD-näytön ohjausfunktiot kirjoitettiin pääosin itse käyttämällä periaatteellisena apuna Adafruit Industries:n ST7565 LCD-kirjastoa [7].

LCD-näytön ohjausta varten mikrokontrollerissa on näytön sisältö tallennettu puskurimuuttujaan *lcd_buffer*. Itse LCD-näyttö päivitetään kopioimalla muuttujan koko sisältö näytön puskuriin kerralla. Ohjausta olisi mahdollista parantaa toimimaan niin, että vain niin kutsutut likaiset pikselit, eli sellaiset pikselit, joihin on tehty muutoksia, päivitetään. LCD-näytön valmistaja suosittelee koko näytön päivittämistä tasaisin väliajoin mahdollisten kontrastierojen välttämiseksi, joten yksityiskohtaista päivitystapaa ei käytetä.

Näytön ohjauksessa on käytössä neljä eri funktiota ja samalla abstraktiotasoa. Kaikki funktiot muokkaavat mikrokontrollerin sisäistä puskurimuuttujaa *lcd_buffer*. Pienimmän abstraktiotason toiminnan toteuttaa funktio *lcd_set_pixel*, jonka argumentteja ovat x- ja y-koordinaatti ja muuttuja *pixel_status*. Funktio muuttaa koordinaattien määräämän pikselin tilaa niin, että kun *pixel_status* on 0, pikseli sammutetaan. Vastaavasti arvolla 1 pikseli sytytetään. Seuraavan tason funktio on *lcd_draw_column:*

```
void lcd_draw_column(uint8_t x, uint8_t y, uint8_t data) {
        uint8_t row;
        for (row = 0; row < 8; row++,y++) {
                if ((data & (1<<(row))) != 0) {
                        lcd_set_pixel(x, y, 1);
                } else {
                        lcd_set_pixel(x, y, 0);
                }
        }
}
```

Funktion argumentteina ovat x- ja y-koordinaatit ja muuttuja *data*, joka sisältää yhden 8-bittisen luvun. Funktio käy muuttujan *data* jokaisen bitin läpi ja *lcd_set_pixel*-funktion avulla piirtää näytölle bitin arvon. Y-koordinaattia kasvatetaan yhdellä jokaisen bitin piirtämisen jälkeen. Seuraavan tason funktio on *lcd_draw_char* (ohjelma 5.1), joka piirtää näytölle yhden kokonaisen merkin. Funktion argumentteina ovat x- ja y-koordinaatti sekä muuttuja *c*, joka sisältää piirrettävän merkin. Funktio suorittaa kaksi eri toimenpidettä. Ensin piirrettävän merkin data haetaan fonttimuuttujasta, minkä jälkeen merkki piirretään puskuriin. Ensimmäisenä funktio määrittää piirrettävän merkin ascii koodin muuttamalla char-tyyppisen muuttujan kokonaisluvuksi. Tulos tallennetaan muuttujaan *w*. Jos arvo osuu välille 32 – 126, löytyy sille fonttikirjastosta vastine. Muussa tapauksessa piirretään virhettä kuvaava vakioarvo. Muuttujaan *addr* haetaan muistiosoite, jossa piirrettävä merkki sijaitsee muuttujassa *font*. Tämä muuttuja on 96x7 – kokoinen taulukkomuuttuja, johon on tallennettu tarvittavien kirjaimien, merkkien ja numeroiden piirtoon tarvittavat

pikseliarvot. Ennen piirtämisen alkua muuttuja *w* nollataan, jotta siihen voidaan tallentaa piirretyn merkin leveys. Seuraavassa vaiheessa funktio hakee aiemmin määritellyn muistiosoitteen avulla piirrettävän merkin tiedot rivi kerrallaan ja piirtää ne *lcd_draw_column*-funktion avulla puskuriin. Lopuksi funktio palauttaa piirretyn merkin leveyden.

```
uint8_t lcd_draw_char(uint8_t x, uint8_t y, uint8_t c) {
        uint8_t w = (int)c;
        uint16_t addr = 0;
        uint8_t data = 0x00;

        if (w < 32 || w > 126) {
                addr = (uint16_t)(&font['#'-0x20]);
        } else {
                addr = (uint16_t)(&font[c-0x20]);
        }

        w = 0;

        while (data != 0xAA) {
                data = pgm_read_byte(addr);
                if (data != 0xAA) {
                        lcd_draw_column(x, y, data);
                }
                ++addr;
                ++x;
                ++w;
        }
        if (w > 7) w = 7;
        return (w);
}
```

Ohjelma 5.1 *LCD-näytön funktio, joka piirtää kokonaisen merkin.*

Korkeimpana abstraktiotasona toimii funktio *lcd_draw_string*, joka piirtää kokonaisen sanan ja käyttää siihen funktiota *lcd_draw_char*.

Loput LCD-näytön ohjaukseen tarkoitetut funktiot käyttävät näitä perusfunktiota ja suorittavat tälle laitteelle yksilöityjä toimintoja. Tällaisia toimintoja ovat esimerkiksi suuren numeron piirto, GPS-logon piirto ja rivikohtainen näytön tyhjennys. Käyttöliittymän piirto suoritetaan myös oman funktionsa avulla. Erilaiset näyttötilat määritetään muuttujien *display* ja *page* avulla.

5.3.4 Akkujännite

Akkujännite mitataan yksinkertaisen vastusjaon ja mikrokontrollerin sisäisen AD-muuntimen avulla. Normaalitilanteessa vastusjako on virran säästämiseksi irrotettuna vastusjaosta ja kytketään kiinni vain mittausta varten. Kun akkujännite on transistorin

avulla kytketty vastusjakoon, suoritetaan AD-muunnos kaksi kertaa. Vastusjako on mitoitettu niin, että 4,8 V akkujännite tuottaa muunnoksesta maksimituloksen, eli 1023. Akkujännitteen tarkastelun tarkkuudeksi riittää 0,1 V, joten tulos skaalataan niin, että akkujännitteen maksimiarvo on 48, joka vastaa jännitettä 4,8V

Mitatun jännitteen perusteella määritellään akun tila, josta kertoo muuttuja *battery_status*. Muuttuja on taulukkomuotoinen ja siihen tallennetaan kolmen perättäisen mittauksen tulokset. Taulukossa 5.6 on esitetty muuttujan eri arvot, niitä vastaavat akkujännitteet ja akun tila. Tilamuuttujan perusteella näytölle piirretään kappaleessa 5.4 kuvattu ilmaisin. Ilmaisimen tilaa muutetaan vain, jos akun tila on ollut sama kolmessa peräkkäisessä mittauksessa. Tällä estetään ilmaisimen turha vaihtelu jännitteen raja-arvojen kohdalla. Tyhjän akun rajaksi on määritetty 3,5 V. Litiumakun tapauksessa on tärkeää, että akun jännite ei pääse laskemaan liian alas, jotta akun turvallinen lataaminen on mahdollista. Laitteen kuluttama virta on aina milliampeereja, joten raja on määritetty alhaiseksi. Lisäksi laite sammuttaa itsensä automaattisesti, jos akkujännite laskee liian alas.

Taulukko 5.6 *Akun tilamuuttuja, jännite ja tila.*

Status	Jännite U	Tila
0	U ≤ 3,3	off
1	3,3 < U ≤ 3,5	0 %
2	3,5 < U ≤ 3,7	33 %
3	3,7 < U ≤ 3,9	66 %
4	U > 3,9	100 %

5.3.5 Virranhallinta

Normaalin toiminnan aikana laitteen akku on aina kytkettynä eikä käyttäjä pääse sitä irrottamaan ilman kotelon avaamista. Kun käyttäjä sulkee laitteen valitsemalla käyttöliittymästä *power off,* siirtyy laite mahdollisimman syvään virransäästötilaan. Käytännössä tämä tarkoittaa sitä, että kaikki ulkoiset laitteet kalenteripiiriä lukuun ottamatta sammutetaan ja mikrokontrolleri siirtyy syvimpään mahdolliseen virransäästötilaan, josta se voidaan vielä herättää ulkoisen keskeytyksen avulla. Ohjelmiston kannalta laitteen sammuttamiseen liittyy joitakin huomioitavia asioita.

Normaalin käytön aikana mikrokontrolleri käyttää vahtikoira-ajastinta (*engl. watchdog*), joka resetoi laitteen, jos ajastimen laskuria ei nollata noin neljän sekunnin kuluessa. Normaalitilanteessa laskuri nollataan jokaisen pääohjelman kierroksen aikana, jolloin

nollausväli on enintään yksi sekunti. Sammutustilanteessa vahtikoira on tärkeää poistaa käytöstä, jotta vahtikoira ei resetoi kontrolleria turhaan, kun laite ei ole käytössä.

Kalenteripiirin tehtävänä on sammutustilanteessa toimia reaaliaikakellona. Normaalin käytön aikana kalenteripiiriä käytetään ajastimena ja reaaliaika on tallennettuna mikrokontrolleriin. Ennen sammutusta on kellonaika siirrettävä takaisin kalenteripiiriin. Kellonaika tahdistetaan kalenteripiirin 1 Hz - signaalin avulla. Kalenteripiirissä tämä signaali on synkroninen piirin sisäisen 32768 Hz kiteen kanssa, jolla piiri laskee aikaa. Kellonajan siirto tahdistetaan sammutustilanteessa niin, että ohjelma jää odottamaan kellosignaalin laskua ja ajan siirto suoritetaan välittömästi signaalin laskevalla reunalla. Näin reaaliaika pysyy mahdollisimman hyvin muuttumattomana.

Kellonaika on kalenteripiirissä tallennettu BCD-muotoon. Mikrokontrollerissa kellonaikaa käytetään kokonaislukumuodossa, jotta ohjelman ajon aikana ei tarvita jatkuvia muunnoksia. Kalenteripiiriin siirron yhteydessä kellonaika muunnetaan takaisin BCD-muotoon. Muunnos suoritetaan kolmessa vaiheessa yksinkertaisilla komennoilla:

```
data[0] = TOD_sec/10;
data[0] = data[0] << 4;
data[0] |= TOD_sec%10;
```

Ohjelmakoodissa muuttujaan *data[0]* tallennetaan sekunnit BCD-muodossa niin, että neljä ylintä bittiä muodostavat sekuntien kymmenet ja neljä alinta bittiä sekuntien ykköset. Kokonaislukumuotoiset sekunnit luetaan muuttujasta *TOD_sec*. Ajan siirtämisen jälkeen kalenteripiirin kello- ja keskeytyssignaali poistetaan käytöstä, jolloin mikrokontrolleri herää vain nappien painallukseen. LCD-näytölle ajetaan sulkukomennot ja näytön virrat katkaistaan kokonaan transistorien avulla.

Laitteen käynnistymiseen tarvittavat rutiinit ovat pääasiassa sammuttamiseen nähden suoraan käänteisiä. Laitteet ja toiminnot, jotka on aiemmin suljettu ja poistettu käytöstä, käynnistetään ja otetaan käyttöön. Erikoisuutena käynnistyshetkellä on akkujännitteen tarkastus. Jos akun jännite on liian alhainen, piirretään näytölle ilmoitus *Battery low! Charge battery.* Ilmoitus näkyy kaksi sekuntia ja laite sammuu.

Käynnistyksen yhteydessä aika siirretään samalla tavalla tahdistetusti mikrokontrollerin muistiin. BCD-muotoinen aika muunnetaan kokonaislukumuotoon komennolla:

```
uint8_t TOD_sec = (data[0] >> 4)*10 + (data[0] & 0x0F);
```

Komennossa muuttuja *data[0]* sisältää sekunnit BCD-muodossa. Desimaaleiksi muutettu aika tallennetaan muuttujaan *TOD_sec*.

5.4 Käyttöliittymä

Laitteen toimintaa ohjataan neljän painonapin avulla. Kuvassa 5.5 on esitetty laitteen pääsivu, joka avautuu, kun laite käynnistetään. Näytön alareunassa olevat tekstit kuvaavat toimintoa, joka kutakin nappia painamalla suoritetaan. Käyttöliittymän vasemmassa yläkulmassa on palkki, joka kuvaa akun varaustasoa. Varaustaso esitetään neljän erilaisen kuvan avulla. Kuvassa 5.5 näkyvä palkki kuvaa täynnä olevaa akkua, muut vaihtoehdot ovat 2/3 täynnä, 1/3 täynnä ja tyhjä. Akkupalkin oikealla puolella oleva tila on varattu SD-kortin virheilmoitukselle. Jos SD-kortti on asennettu, mutta sen alustus epäonnistuu, tulostuu tähän kohtaan viesti *E!:SD*. Näytössä keskellä ylhäällä ilmoitetaan kellonaika. Kellonajan oikealla puolella oleva symboli kuvaa GPS-yhteyden tilaa. Ilmoitukselle on neljä erilaista vaihtoehtoa:

- Off, GPS on pois käytöstä
- Wait, satelliitteja etsitään
- 3/5, paikannus onnistuu ja satelliitteja on seurannassa alle kahdeksan
- OK!, paikannut onnistuu ja satelliitteja on seurannassa kahdeksan tai enemmän

Kuva 5.5 *Käyttöliittymän pääsivu.*

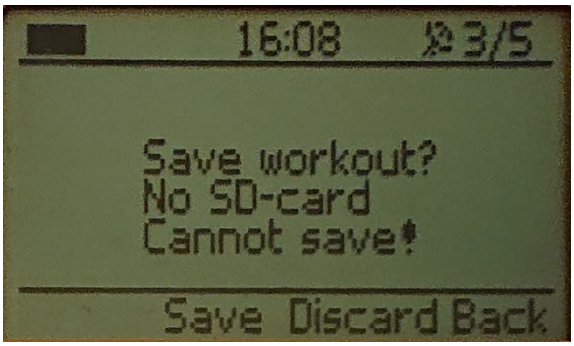
Näytön keskiosassa suuremmalla fontilla esitetyt tiedot ovat nopeus kilometreinä tunnissa, matka kilometreinä, mitattu aika esitystavalla *h:min:s* ja lämpötila celsiusasteina. Start-toiminto käynnistää ajanoton, matkan mittauksen ja nousu- ja laskumetrien tallennuksen. Nappia painettaessa kuuluu lyhyt äänimerkki, jotta käyttäjä voi varmistua painalluksen rekisteröinnistä. Stop-toiminto pysäyttää vastaavasti

ajanoton, matkan mittauksen ja nousu- ja laskumetrien tallennuksen. Myös tämä toiminto vahvistetaan äänimerkillä.

Page-nappia painamalla päästään käyttöliittymän toiselle sivulle (kuva 5.6). Tällä sivulla näytön keskiosassa on esitetty keskinopeus kilometreinä tunnissa (*engl. average speed*), nousumetrit (*engl. ascend*) ja laskumetrit (*engl. descend*). Painamalla page-nappia uudestaan palataan pääsivulle.

Kuva 5.6 *Käyttöliittymän toinen sivu.*

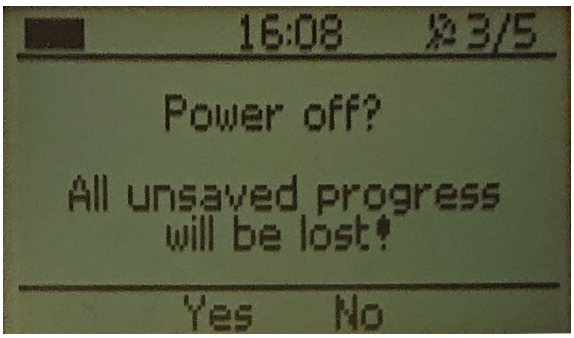
Kun mittari on stop-tilassa, on siihen mennessä kerätyt tiedot mahdollista nollata tai tallentaa SD-kortille. Kun käyttäjä pitää stop-painiketta pohjassa, vaihtuu käyttöliittymä tallennusnäkymään (kuva 5.7). Jos SD-korttia ei ole asennettu, ilmoitetaan siitä tekstillä *No SD-card. Cannot save!*. Tässä tapauksessa save-toiminto ei tee mitään. Virheilmoitus poistuu, kun SD-kortti on asennettu. Save-toiminto tallentaa SD-kortille päivämäärän, ajan, matkan, keskinopeuden, nousumetrit ja laskumetrit. Txt-muotoisen tiedoston nimi koostuu päivämäärästä ja päiväkohtaisesta juoksevasta numerosta. Tallennuksen jälkeen kaikki tiedot nollataan. Discard-toiminto nollaa kaikki tiedot ilman tallennusta. Nollaus kuitataan tekstillä *Discarding* ja kahdella lyhyellä äänimerkillä. Valikosta palataan päävalikkoon painamalla back.



Kuva 5.7 *Käyttöliittymän tallennusnäkymä.*

Päävalikossa pitämällä pohjassa power-nappia päästään sammutusvalikkoon (kuva 5.8). Painamalla Yes kuuluu pitkä yhtenäinen äänimerkki ja laite sammuu. Samalla kaikki

tiedot nollataan. Painamalla No palataan takaisin päävalikkoon. Sammuneen oleva laite käynnistyy pitämällä power-nappia pohjassa.



Kuva 5.8 *Käyttöliittymän sammutusnäkymä.*

6. TESTAUS JA TULOKSET

Laitteen testaus jakautuu kahteen osaan. Ensimmäinen suoritettiin ohjelmistokehityksen alussa ja koski yksittäisiä komponentteja ja niiden toimintaa. Toisessa vaiheessa testattiin käyttökunnossa olevaa laitetta.

6.1 Komponenttikohtainen testaus

Ensimmäinen testattava komponentti oli mikroprosessori. Ohjelmointilaite kytkettiin kiinni levyllä olevaan liittimeen ja Atmel Studio:lla luettiin prosessorin tyyppi ja ns. sulakebittien arvot. Yhteys prosessoriin todettiin toimivaksi ja konfiguraatio oikeaksi. Samalla asetettiin sulakebittien avulla käyttöön sisäinen oskillaattori täydellä 8 MHz kellotaajuudella. Myös alhaisen käyttöjännitteen seuranta (*engl. brown-out detection*) kytkettiin päälle 2,7 V rajajännitteellä. Jos käyttöjännite laskee tämän arvon alle, prosessori resetoi itsensä.

Testausta jatkettiin LCD-näytöstä. Toimiva näyttö oli testauksen ja ohjelmistokehityksen kannalta olennaisin osa, koska sitä voitiin käyttää apuna muiden komponenttien testauksessa. LCD-näyttö testattiin ajamalla näytölle valmistajan ohjeen mukaisesti alustusruutiini ja kytkemällä näytön kaikki pisteet päälle. Näyttö toimi heti ensimmäisellä testauksella oikein. Ennen testauksen jatkamista näytönohjauksen ohjelmisto viimeisteltiin, jotta näytölle saatiin piirrettyä informaatiota muita testejä varten.

Seuraava kohteena oli painonapit. Kullekin napille ohjelmoitiin erilaisia LCD-näyttöä ohjaavia toimintoja, kuten näytön käynnistys ja sammutus ja kaikkien pikseleiden näyttö ja sammutus. Neljästä painonapista kolme todettiin toimiviksi. Yksi nappi ei reagoinut mitenkään. Mikroprosessorin porttikonfiguraatio tarkastettiin ensin ja todettiin oikeaksi. Kaikki nappien pinnit oli määritetty sisääntuloiksi ja niillä oli käytössä sisäiset ylösvetovastukset. Yleismittarin avulla itse painonappi todettiin toimivaksi, mutta signaali ei kulkenut mikroprosessorille asti. Vian aiheutti debounce-piirin yhden jalan huono juotos. Juotoksen korjauksen jälkeen kaikki napit toimivat.

Painonappien jälkeen testattiin summerin toiminta. Summerille syötettiin 4 kHz kanttiaaltoa niin, että painonappi painamalla ääni saatiin käynnistettyä ja sammutettua. Summeri todettiin toimivaksi ja äänenvoimakkuus korvakuulolla riittäväksi.

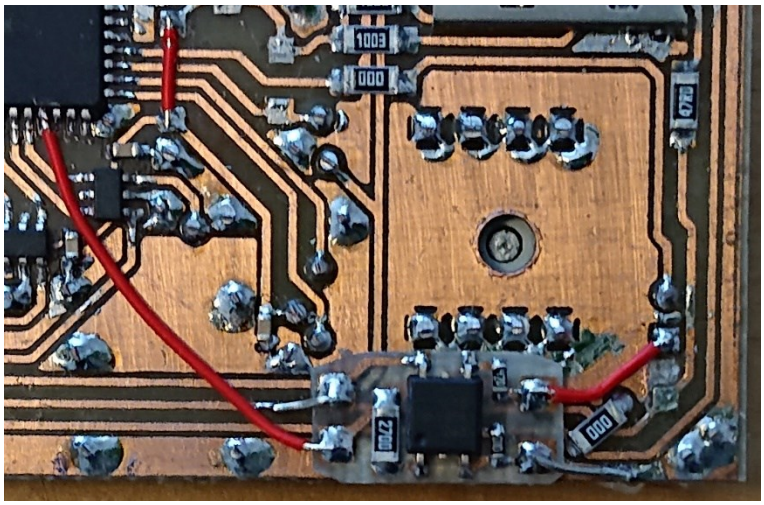
Seuraava testattava komponentti oli paineanturi. Anturi on hyvin yksinkertainen ja kytkentöinä on käyttöjännitteen lisäksi vain liittymätyypin valinta ja SPI-liitynnän kolme nastaa. Anturin testaus suoritettiin kokonaan SPI-väylän avulla. Anturille syötettiin

datalehden mukaiset komennot, jolla piiri suoritti lämpötilan mitatun arvon AD-muunnoksen. Arvo luettiin piiristä ja tulostettiin LCD-näytölle. Lämpötila valittiin testauksen kohteeksi, koska sen oikeellisuus oli helppo todeta. Ensin näytti siltä, että lämpötila näkyy oikein. Testausta jatkettiin puhaltamalla lämmintä ilmaa anturille, jolloin lämpötilan olisi pitänyt nousta. Arvossa ei kuitenkaan tapahtunut muutosta. Yleismittarin avulla mitattiin liikennettä väylällä ja todettiin, että kun piiri lähetti tietoa, oli väylä koko ajan lepotilan arvossaan, joka oli ylösvetovastusten määräämä 1. Sattumalta tämä arvo tuottaa laskennassa lähes huonelämpöä vastaavan arvon. Käytännössä piirin SPI-väylälle lähettävä nasta ei saanut kontaktia piirilevyyn. Liitos kuumennettiin uudestaan ja tinaa lisättiin, jolloin liikenne alkoi kulkea ja lämpötila näkyi oikein.

Jäljellä olevista kolmesta komponentista seuraavaksi testattiin kalenteripiiri. Piirissä on käyttöjännitteen ja väyläliitynnän lisäksi kaksi ulostuloa, jotka on kytketty mikroprosessorin sisääntulonastoihin. Toinen näistä ulostuloista on kellosignaali ja toinen keskeytys. Piiri testattiin lähettämällä SPI-väylän kautta komennot, joilla kellosignaali lähti käyntiin 1 Hz taajuudella ja vastaavasti keskeytys tapahtui samalla taajuudella. Mikroprosessorin sisääntulojen tila tulostettiin LCD-näytölle ja kalenteripiiri todettiin toimivaksi.

Viimeinen piirilevylle juotettu testattava komponentti oli GPS-moduuli. Moduuli oli datalehden mukaan erityisen herkkä oikeanlaiselle virran päälle ja poiskytkennälle, joten moduuli juotettiin kiinni levylle vasta kun tiedettiin, että kaikki muu toimii. Moduulissa on kaksi erillistä nastaa, joilla määritetään UART-väylän nopeus ja tietoliikennetyyppi. Datalehden mukaan näitä ei SPI-käytössä tarvittu. GPS-moduuli ei ota vastaan komentoja SPI-väylän kautta, joten testaus suoritettiin yksinkertaisesti kytkemällä moduuli päälle, seuraamalla liikennettä väylällä ja näyttämällä data LCD-näytöllä. Oletusarvona oli, että jokainen vastaanotettu tavu sisältää yhden merkin, joista muodostuu NMEA-standardin mukainen lause. Näytölle tulostui kuitenkin täysin epäloogisia merkkejä eikä seassa ollut juuri ollenkaan luettavia kirjaimia. Vika olisi voinut olla väylän häiriöissä, mutta koska kaikki muut komponentit toimivat ongelmitta etsittiin vikaa muualta. Lopulta apua kysyttiin piirin valmistajalta, josta vastaus saapuikin nopeasti. Datalehden tiedoista huolimatta myös SPI-väylää käytettäessä täytyy piirin UART-väylän määritysnastoihin kytkeä vastukset. Vastukset lisättiin, jolloin piiriltä saatiin ulos selviä kirjaimia ja lopulta NMEA-lauseita. Tässä vaiheessa testausta huomattiin kuitenkin, että GPS-moduulin levylle juottamisen jälkeen väyläliikenne paineanturilta ja kalenteripiiriltä mikroprosessorin suuntaan loppui kokonaan. Syy tähän löytyi hyvin nopeasti yleismittarin avulla. Todettiin, että GPS-piiri ei vapauta väylää kelluvaksi vaan vetää sen vahvasti alas, vaikka piiri ei olekaan valittuna. Myös tästä oltiin yhteydessä

valmistajaan ja saatiin vahvistus, että moduulin käyttämä SiRFstar® GSD4e™ -piiri toimii näin rautatasolla eikä asialle voi tehdä mitään. Käytännössä ainoaksi mahdollisuudeksi jäi fyysisesti irroittaa GPS-moduulin nasta väylältä. Piirilevylle lisättiin tätä tarkoitusta varten mosfet-rele (kuva 6.1), jolle otettiin ohjaus mikroprosessorilta. Releen sisällä on led, joka kytkee fotosensitiivisen transistorin johtavaan ja johtamattomaan tilaan. Relekytkentä todettiin toimivaksi ja SPI-väylän liikenne alkoi taas kulkea normaalisti.



Kuva 6.1 *Piirilevylle lisätty mosfet-rele.*

SD-kortin toimintaa ei tässä vaiheessa testattu tarkemmin. Kortin liittimen kontaktit varmistettiin mittaamalla.

6.2 Ohjelmiston testaus

Suurin osa ohjelmiston testauksesta oli toiminnallista testausta, joka on käsitelty kappaleessa 6.3. Ohjelmiston ja etenkin käyttöliittymän toimintaa testattiin kuitenkin myös erikseen ennen laitteen kotelointia. Testauksella varmistettiin ohjelmiston nopea toiminta ja haluttiin korjata mahdolliset jumittumiseen johtaneet virheet ennen toiminnallista testausta.

Ensimmäisenä testattiin käyttöliittymän reagointinopeus. Tavoitteena oli, että ohjelmisto toimii ilman viiveitä, kun käyttäjä painaa painikkeita. Käytännössä aikakriittisinä toimintoina testattiin mittauksen käynnistys ja pysäytys sekä sivunvaihto. Sivunvaihto vaikutti alustavissa testeissä toimivan hyvin. Sivu vaihtui välittömästi painikkeen painalluksen jälkeen ja toiminta tuntui sulavalta. Testauksen aikana nappia paineltiin erilaisissa käyttötilanteissa ja vaikka itse sivunvaihto toimikin oikein, toi testaus esille toisenlaisen virheen. Toisinaan napin painamisen jälkeen ajan mittaus saattoi pysähtyä, GPS-tiedot lakata kulkemasta tai koko ohjelma jumiutua. Yhteiseksi tekijäksi näille

virheille löytyi nopeasti SPI-väylä. Ohjelmiston rakenteessa ei oltu huomioitu napin aiheuttamaa keskeytystä riittävän hyvin, jolloin keskeytys saattoi tulla, kun SPI-väylä oli varattuna. Ohjelma ei jäänyt odottamaan väylän vapautumista vaan nykyinen toiminta keskeytyi ja uusi korvasi sen, jolloin edellinen SPI-viesti jäi kesken. Ongelma korjattiin estämällä keskeytyksen suoritus, kun SPI-väylällä on liikennettä. Muutos korjasi sekoiluongelmat, mutta hyvin nopeasti tuli ilmi, että sivunvaihto ei enää toiminut nopeasti. Sivu saattoi vaihtua heti tai vasta viiveellä. Viive johtui siitä, että ohjelma jäi odottamaan SPI-väylän vapautumista. Yleisesti väyläliikenne laitteessa on niin nopeaa ja lyhytaikaista, että viiveitä ei pitäisi ihmisen pystyä havaitsemaan. GPS-piirin tapauksessa tietoa kuitenkin siirretään enemmän ja sitä myös käsitellään siirron jälkeen. Ohjelma jäi odottamaan myös käsittelyn valmistumista, mikä aiheutti käyttäjälle asti näkyvän viiveen. Ohjelmaa korjattiin niin, että sivunvaihto oli mahdollista myös käsittelyn aikana, jolloin viive poistui.

Sivunvaihdon testauksen yhteydessä havaittiin myös nousu- ja laskumetreihin liittyvä virhe. Vaikka mittari oli paikallaan sisätiloissa, kertyi sekä nousu- että laskumetrejä tasaiseen tahtiin. Muutamassa minuutissa metrejä kertyi kymmeniä. Vikaa lähdettiin etsimään korkeuden laskentaan käytetyistä kaavoista. Itse paineen muunnos korkeudeksi toimi selvästi oikein ja vaikka ilmanpaineen laskennassa tarvitaan hyvin raskasta laskentaa, todettiin senkin toimivan oikein. Lopulta ilmanpaineen arvo piirrettiin näytölle tarkastelua varten. Näytöltä saattoi huomata, että paineen arvo vaihteli pahimmillaan noin 0,04 mbar. Anturipiiri oli sijoitettu piirilevylle erilleen muista ja datalehden ohjeen mukaisesti SPI-väylä pidettiin lepotilassa AD-muunnoksen aikana, joten selvää syytä vaihtelulle ei löytynyt. Piiriin itseensä kohdistuva häiriösignaali on kaikesta huolimatta mahdollinen, mutta vikaa lähdettiin korjaamaan ohjelmallisesti. Kipinän tähän antoi referenssilaitteena testauksessa käytetty Garmin Edge 130. Garminin huoltovalikon kautta pystyi tarkastelemaan ilmanpaineen raakadataa ja kävi ilmi, että datassa on täsmälleen samanlaista heilahtelua, kun laite on paikallaan. Vikaa pidettiin ensin yksinkertaisena ja laskennassa otettiin käyttöön raja-arvo, jonka alittavat korkeuden muutoksen arvot jätettiin huomioimatta. Lisäksi lämpötilan ja paineen arvon mittausta muutettiin niin, että AD-muunnos suoritettiin 10 kertaa ja mittaustulosten keskiarvoa käytettiin laskennassa. Tällä muutoksella nousu- ja laskumetrejä ei kertynyt paikallaan ollessa. Ratkaisu ei kuitenkaan ollut muilta osin toimiva, kuten jatkotestaus kappaleessa 6.3 osoitti.

Mittauksen käynnistymisen ja pysäytyksen osalta ohjelma toimi yleensä oikein, mutta välillä tuli tilanteita, joissa painallus tuli merkkiäänen perusteella rekisteröidyksi, mutta ajastin ei joko käynnistynyt tai pysähtynyt. Ajan mittaus tapahtuu ulkopuolisessa

kalenteripiirissä, jota ohjataan SPI-väylän avulla. Mittauksen tilaa muutetaan piirin osoitteessa *00h* sijaitsevan rekisterin (kuva 6.2) arvoa *STOP* muuttamalla. Vastaavasti mittauksen tila voidaan tarkastaa lukemalla saman bitin arvo.

Bit	7	6	5	4	3	2	1	0
Symbol	EXT_ TEST	T	STOP	TSF1	POR_ OVRD	12_24	MI	SI
Reset value	0	0	0	0	1	0	0	0

Kuva 6.2 *Kalenteripiirin rekisteri 00h. [12]*

Vikaa lähdettiin selvittämään lukemalla *STOP*-bitin arvo käynnistys- ja sammutusyrityksen jälkeen ja tulostamalla se LCD-näytölle. Samalla muutettiin ja luettiin *12_24*-bitin tila. Tällä bitillä valitaan 12 ja 24 tunnin kellon väliltä. Ylimääräisellä muutoksella varmistettiin komennon perille meno. Testissä kävi ilmi, että *12_24*-bitti muuttui aina oikein, mutta välillä *STOP*-bitti ei muuttanut tilaansa. Käsky meni selvästi perille, mutta jostain syystä se ei aina toiminut. Kalenteripiirin datalehden mukaan, kun rekistereihin kirjoitetaan tai niistä luetaan, pysäyttää piiri automaattisesti kaikkien laskurien sisällön, joten käskyjen ajoituksella ei ole merkitystä [12]. Virheelle ei löytynyt selvää syytä, ja se korjattiin muuttamalla ohjelmaa niin, että *STOP*-bitin tila tarkastettiin jokaisen muutoksen jälkeen ja jos tila ei muuttunut, annettiin muutoskomento uudestaan. Muutoksen jälkeen käynnistys ja pysäytys toimivat oikein.

Ohjelmiston testauksen osalta seuraavana kohteena oli ohjelman tahdistus. Ohjelma tahdistettiin kalenteripiirin syöttämän 1 Hz -kellosignaalin avulla niin, että mikroprosessori herää sekunnin välein, suorittaa tarvittavat mittaukset ja siirtyy heti syvään virransäästötilaan. Kellosignaalin lisäksi vain painikkeiden painaminen herättää laitteen. Tahdistus todettiin heti hyvin ja oikein toimivaksi. Ongelmaksi muodostui kuitenkin mitatun ajan näytölle piirto, joka laahasi jäljessä. Aiheuttajaksi selvisi kalenteripiirin rakenne, jossa 1 Hz -kellosignaali ei toimi synkronisesti ajan mittauksen kanssa. Piirissä oli kuitenkin saatavilla myös ohjelmoitava keskeytyspinni, joka toimi samassa tahdissa ajan mittauksen kanssa. Keskeytys otettiin käyttöön ja liitettiin mikrokontrolleriin niin, että kontrolleri herää myös ajan mittauksen tuottaman 1 Hz -keskeytyksen tahdissa ja tämä keskeytys päivittää näytön. Näin numerot vaihtuvat näytölle oikeaan aikaan. Keskeytyksen käyttöön ottaminen aiheutti kuitenkin ongelmia. Mikrokontrollerin portti, johon keskeytys kytkettiin ei pysty tunnistamaan portin tilan muutosta vaan keskeytys aiheutuu, kun portti on joko ylhäällä tai alhaalla riittävän pitkään. Tässä toimintatavassa keskeytysvektori suoritetaan uudestaan niin pitkään, kunnes portin tila vaihtuu. Kalenteripiirin keskeytykseksi oli valittu pulssi, jonka pituutta ei voinut säätää, joten keskeytysvektori tuli suoritetuksi lukemattomia kertoja koko

pulssin pituuden ajan. Toimintaa muutettiin niin, että kalenteripiirin keskeytyslippu nollattiin aina keskeytyksen tapahduttua, joten keskeytys tuli suoritetuksi vain kerran. Samalla huomattiin, että vastaava tilanne on mahdollinen myös 1 Hz -kellosignaalin kanssa. Ohjelma oli rakenteeltaan tehty sellaiseksi, että vaikka keskeytysvektori suoritettiinkin lukemattomia kertoja, ei se näkynyt ohjelman suorituksessa kuin mahdollisena hidasteluna. Tilanne korjattiin ottamalla mikrokontrollerin INT6-keskeytys, johon kellosignaali on kytketty, pois käytöstä aina, kun keskeytys suoritetaan. Keskeytys palautetaan ohjelmassa käyttöön, kun kellosignaali on vaihtanut tilaansa, jolloin ongelmaa ei synny.

Edellä mainittu tahdistus lisättiin myös laitteen käynnistysfunktioon niin, että kun kellonaika luetaan kalenteripiiristä mikrokontrollerin muistiin, suoritetaan tiedonsiirto kalenteripiirin tuottaman keskeytyksen tahdistamana, jolloin reaaliaika pysyy oikeana.

Viimeisenä testattavana kohteena oli SD-kortille kirjoitus. Kortin alustuksessa ja tiedostojärjestelmän hallinnassa käytetty valmis FatFs-moduuli toimi odotusten mukaisesti heti oikein. Kortille kirjoitettavien tietojen tallennus suoritettiin kuitenkin itse kirjoitetulla funktiolla, jonka toiminnassa havaittiin testin perusteella pieniä puutteita. Tekstitiedosto, johon tarvittavat mittaustulokset kirjoitettiin, avautui SD-kortille normaalisti ja kaikki muut tiedot siirtyivät ilman virheitä, mutta aika oli virheellinen. Esimerkkinä aika 35 min 17 s näkyi SD-kortilla muodossa 114 h 107 min 114 s. Kyseessä oli yksinkertainen kirjoitusvirhe. Aika haettiin kalenteripiiristä BCD-muodossa ja tallennettiin muuttujaan *data* niin, että *data[0]* sisälsi sekunnit, *data[1]* minuutit ja *data[2]* tunnit. Ohjelmakoodi luki virheellisesti tietoja vääristä kohdista muuttujaa, mikä johti ylivuotoon ja vääriin numeroihin. Ohjelma korjattiin muotoon:

```
sprintf(buffer,"Aika: %u%uh %u%umin %u%us\r\n",data[2]>>4,
data[2]&0x0F,data[1]>>4,data[1]&0x0F,data[0]>>4,data[0]&0x0F);
```

Tämä käsky kirjoittaa muuttujaan *buffer* selkokieliset tekstit *Aika:*, *h*, *min* ja *s*. Käskyssä näkyvät *%u* – merkinnät korvautuvat järjestyksessä käskyn lopussa näkyvien muuttujien sisällöllä. Esimerkiksi ensimmäinen *%u* korvautuu muuttujalla *data[2]*, jonka arvoa on siirretty neljä bittiä oikealle, jolloin jäljelle jää neljä ylintä bittiä. Nämä bitit kuvaavat tuntien kymmeniä.

6.3 Toiminnallinen testaus

Laitteen toiminnan testaus aloitettiin siinä vaiheessa, kun laitteen kotelo oli valmis ja ohjelmisto sisälsi kaikki toiminnot. Testausta varten laite kiinnitettiin polkupyörään ja ajettiin erilaisia testiajoja. Suurin osa testeistä suoritettiin ajamalla n. 11,7 km mittaista

työmatkaa, joka sisälsi sopivasti vaihtelevia korkeuksia ja sekä suoraa että mutkittelevaa ajoa. Vertailulaitteena käytettiin Garmin Edge 130 -polkupyörätietokonetta, jossa on GPS-mittaus ja erillinen ilmanpaineanturi. Testiajojen jälkeen molempien laitteiden mittaamat tiedot tallennettiin tarkastelua varten. Mahdolliset muut huomiot kirjoitettiin ajon jälkeen muistilapulle. Testaus suoritettiin yhden kuukauden pituisen jakson aikana niin, että testauksessa havaitut virheet pyrittiin korjaamaan ennen seuraavaa testausta. Taulukossa 6.1 on esitetty yhteenveto testauksen aikana havaituista virheistä.

Taulukko 6.1 *Havaitut virheet toiminnallisessa testauksessa.*

Pvm	Havaitut virheet
7.6.2019	Nopeus ei nollaudu stop - tilanteessa, keskinopeus väärin, lämpötila väärin ja matka liian lyhyt.
15.6.2019	Matka liian lyhyt ja mittaus pysähtelee, korkeus 0 m.
21.6.2019	Korkeus 2,4 kertainen, matkamittaus 130m liian lyhyt 11,7 km matkalla.
1.7.2019	Korkeus viidesosa oikeasta.
2.7.2019	Korkeus noin puolet oikeasta.
3.7.2019	Korkeus kymmenesosa oikeasta.
9.7.2019	Korkeus 0 m.

Ensimmäisessä testauksessa havaittiin yhteensä neljä virhettä. Kun mittaus pysäytettiin painamalla stop-painiketta, jäi näytölle viimeisin nopeus, vaikka laite oli paikallaan. Virhe korjattiin ohjelmassa asettamalla nopeutta kuvaava muuttuja nollaksi aina kun mittaus pysäytetään. Myös mittarin näyttämä keskinopeus oli väärin. Keskinopeudeksi tallentui 1,2 km/h, kun oikea arvo oli 17,2 km/h. Poikkeaman aiheutti virhe keskinopeuden laskentakaavassa. Matkaan käytetty aika on tallennettu muuttujaan BCD-muodossa niin, että neljä ylintä bittiä muodostaa kymmenet ja neljä alinta ykköset. Keskinopeuden laskentaa varten aika pitää muuttaa desimaaliluvuksi. Muunnos oli alun perin suoritettu apumuuttujan avulla bittejä siirtämällä, mutta muunnos toimi väärin. Muunnos korjattiin yksinkertaisempaan ja oikein toimivaan muotoon, joka on esitetty kappaleessa 5.3.5.

Ulkolämpötilan arvo näkyi väärin niin, että ulkolämpötilan vaihdellessa välillä 21,0 – 24,0 astetta näkyi näytöllä koko ajan arvoja väliltä 22,0 – 22,9. Väärän lukeman aiheutti ohjelmakoodin virhe, jossa näytölle piirrettiin kymmenet myös ykkösten paikalle, jolloin vaikka lämpötilan mittaus toimi oikein, ei näytölle piirtynyt ollenkaan muuttuvaa ykkösten osuutta. Koodi korjattiin näyttämään ykköset oikein. Viimeisenä virheenä havaittiin liian lyhyt mitattu matka. Tallentunut arvo oli 10,65 km, kun oikea kuljettu matka oli 11,70 km. Syyksi pääteltiin float-muuttujien tarkkuuden loppuminen. AVR-mikroprosessorin maksimitarkkuus liukuluvuille on 7 merkkiä riippumatta pilkun paikasta ja vaikutti siltä, että tästä syystä desimaaleille ei jää tarpeeksi tilaa. Matkan laskennan kaava muutettiin

käyttämään pelkkiä kokonaislukuja ja samalla paikan muutoksen perusteella tehtävän laskennan tarkkuudeksi muutettiin 2 cm.

Toisessa testissä havaittiin, että matkan mittaus antaa edelleen liian lyhyitä tuloksia ja mittaus pysähtelee aika ajoin. Virhettä lähdettiin etsimään käytetyistä laskentakaavoista, joiden tulokset tarkastettiin käsin. Kaikki tulokset todettiin oikeiksi, joten ohjelmiston toimintaa alettiin tutkia vaihe vaiheelta. LCD-näytölle piirrettiin luetut GPS-koordinaatit, niiden muutos asteiksi, asteiden muunnos radiaaneiksi, pisteiden välisen kulman kosiniarvo ja pisteiden välimatka. Laitteen kanssa lähdettiin kävelemään ulos, jotta luvut muuttuisivat hitaasti ja virhe olisi helpompi havaita. Testin aikana havaittiin nopeasti, että kun asteina esitetyn koordinaatin kymmenesosat saivat lukuarvon 7, 8 tai 9, sekosi laskenta täysin. Tällä perusteella ohjelmakoodista löytyi kohta, jossa näillä lukuarvoilla tapahtui ylivuoto. Kaavassa kymmenesosien lukuarvo kerrottiin luvulla 10000. Tämän lukuarvon perusteella kertolasku tulee suoritetuksi 16-bittisenä, jolloin suurin mahdollinen tulos on 65536. Jos kertolasku suoritetaan suuremmilla kymmenesosien arvoilla kuin 6, ei tulos mahdu enää 16-bittiseen muuttujaan ja tapahtuu ylivuoto. Muissa saman kaavan kertolaskuissa ongelmaa ei esiinny, koska laskenta tulee kertoimien perusteella automaattisesti suoritetuksi riittävällä tarkkuudelle. Kaava korjattiin lisäämällä lukuarvon 10000 perään kirjaimet UL, jolloin laskenta suoritetaan *unsigned long* -muuttujatyypillä, eli 32-bittisenä:

```
mm = (coordinate[3]-'0')*1000000 + (coordinate[4]-'0')*100000 + (coordinate[6]-'0')*10000UL + (coordinate[7]-'0')*1000 + (coordinate[8]-'0')*100 + (coordinate[9]-'0')*10;
```

Kaavassa merkkeinä tallennettu koordinaatti muutetaan desimaalimuotoon kokonaisluvuksi niin, että merkkijono *12.3456* muutetaan muotoon 123456. Desimaalipiste on tallennettu muuttujaan *coordinate[5]* ja jätetään laskennassa huomioimatta. Laskennan tulos tallennetaan muuttujaan *mm*.

Korkeuden mittauksen todettiin näyttävän nollaa. Kappaleessa 6.2 mainittu korkeusarvon värähtelyn suodatusarvo todettiin liian korkeaksi, jolloin sen ylittäviä muutoksen arvoja ei tullut ollenkaan, eikä korkeuden muutosta päässyt kumulatiivisesti kertymään. Uudeksi arvoksi asetettiin 400, joka tarkoittaa 40 senttimetriä.

Kolmannessa testissä matkan mittaus oli enää 130 m lyhempi kuin todellinen matka. Virheelle pääteltiin olevan kaksi todennäköistä syytä. Laskennassa ei tässä vaiheessa oltu huomioitu korkeuden muutosta, vaan matka laskettiin linnuntietä pitkin. Laskentaa muutettiin niin, että kuljettu matka laskettiin pythagoraan lauseen avulla linnuntietä kuljetusta matkasta ja korkeuden muutoksesta. Muutosta testattiin erikseen useilla lyhyillä ajoilla, joiden perusteella muutos todettiin toimivaksi. Muutoksen vaikutus matkan

laskentaan on kuitenkin häviävän pieni. Jos kuljettu matka linnuntietä on esimerkiksi 11,7 km ja korkeuden muutos 200 m, saadaan pythagoraan lauseella kuljetuksi matkaksi:

$$\sqrt{(11700\ m)^2 + (200\ m)^2} \approx 11701{,}7\ m$$

Vaikutus 11,7 km matkalla on 1,7 metriä, mikä ei selitä 130 metrin poikkeamaa. Todennäköinen syy on, että matka lasketaan kahden GPS – koordinaatin välisen lyhimmän etäisyyden mukaan. Tällöin matka tulee lasketuksi täsmälleen oikein vain, jos ajetaan suoraan. Mutkittelu ja etenkin 90 asteen käännökset aiheuttavat tilanteen, jossa oikea kuljettu matka on pidempi kuin sekunnin välein määritettyjen sijaintien välinen suora välimatka. Testien aikana pystyi visuaalisesti havaitsemaan, että pitkien suorien osuuksien aikana kumulatiivinen virhe kertyi selvästi hitaammin tai ei ollenkaan. Virhettä voisi pienentää esimerkiksi käyttämällä GPS-datasta saatavaa kulkusuuntatietoa. Kahden pisteen välisen kulkusuunnan muutoksen perusteella matkaan voitaisiin lisätä erikseen määritetty arvo. Ominaisuuden lisääminen vaatisi kuitenkin laajaa testausta korrelaation löytämiseksi ja edelleen laskennan oikean tarkkuuden määrittämiseksi, joten se päätettiin jättää tekemättä, koska se ei kuulu enää tämän diplomityön aihepiiriin. GPS-moduuli tukee myös 5 Hz taajuudella tapahtuvaa paikannusta, mikä todennäköisesti parantaisi tarkkuutta. Tilaa ei kuitenkaan ole mahdollista ottaa käyttöön SPI-väylän kautta.

GPS-mittauksessa on hyvin todennäköisesti samanlaista kohinaa kuin ilmanpaineen mittauksessa. Tämä tuli automaattisesti huomioitua ohjelmassa, jossa matkan laskennassa oli alusta asti käytössä ehto, joka esti matkan kertymisen, kun nopeus on 0. Kohinan vaikutus tuloksiin liikuttaessa ei tullut ilmi testeissä. Samaa ehtoa ei alun perin ollut käytössä ilman paineen mittauksessa, mutta myöhemmin se lisättiin.

Kaikki seuraavat testit keskittyivät korkeuden laskennan testaukseen. Alun perin käytetty lähestymistapa, jossa vain tietyn raja-arvon ylittävät arvot lasketaan, osoittautui useiden testien jälkeen riittämättömäksi. Ehto, jonka mukaan nousu- ja laskumetrejä ei kertynyt ei myöskään korjannut toimintaa liikuttaessa. Ensin näytti siltä, että raja-arvoa säätämällä voitaisiin saavuttaa riittävä tarkkuus, mutta ongelmaksi muodostui väärään suuntaan lasketut korkeusarvot. Pelkkää raja-arvoa säätämällä muodostui tilanne, jossa selvään ylämäkeen ajettaessa nousuarvo kertyi järkevästi, mutta samalla kertyi myös pieni määrä laskumetrejä. Tällainen virhe ei ole hyväksyttävää. Virhettä lähdettiin korjaamaan periaatteella, jossa jokaisen mittauksen kohdalla muutoksen suunta tallennettiin erikseen omaan muuttujaan. Muuttujan perusteella nousu- ja laskumetrin laskettiin vain, jos kolmesta mittauksesta kaksi oli saman suuntaisia. Kaavaan yhdistettiin myös raja-arvotarkastelu, mutta pienemmällä 25 cm arvolla. Heti

ensimmäinen lyhyt testi ylämäkeen osoitti, että tämä laskentatapa ei korjannut virheellistä laskumetrien kertymistä ylämäessä. Laskentaa laajennettiin niin, että tarkasteluun otettiin viisi mittausta, joista kolmen tuli olla samaan suuntaan, jotta metrejä kerrytettiin. Testin perusteella väärään suuntaan kertyneiden metrien määrä väheni selvästi, mutta vastaavasti myös oikeaan suuntaan ei saatu enää millään oikeaa tulosta. Kaavan kanssa ajettiin useita yksittäisiä lyhyitä testiä raja-arvoa säätäen, mutta lopulta metrin lakkasivat kertymästä kokonaan ja lähestymistapaa päätettiin vaihtaa vielä kerran.

Lopulliseksi ratkaisuksi muodostui raja-arvoa toisella tavalla käyttävä kaava. Tällä kertaa mitattua korkeuden muutosta verrattiin raja-arvoon. Jos muutos oli vähintään raja-arvon suuruinen, vähennettiin raja-arvo mittaustuloksesta ja lopputulos tallennettiin. Ohjelmaa muutettiin myös niin, että mittarin huoltovalikosta oli mahdollista muuttaa raja-arvoa testin aikana. Lopulta useiden lyhyiden testien perusteella raja-arvoksi valittiin 24 cm.

Viimeisten testien aikana huomattiin korkeuden mittauksessa vielä yksi virhe. Puuskittainen tuuli aiheutti nousumetreihin jopa satojen metrien suuruisia virheitä. Korkeuden laskentaan lisättiin myös toinen raja-arvo, joka hylkää yli 5 metrin suuruiset muutokset korkeudessa. Sadan kilometrin tuntinopeudella tämä mahdollistaa noin 20 % jyrkkyyden, mikä katsottiin riittäväksi. Laskentaan lisättiin myös laskuri, joka seuraa hylättyjen mittausten määrä. Jos perättäisiä hylkäyksiä on enemmän kuin viisi, siirtää ohjelma mittauksen referenssipisteen uusimman mittauksen kohdalle. Näin ohjelma ei jää jumiin tuulisella säällä.

Viimeisissä testeissä korkeuden nousu- ja laskumetrien kertyminen saatiin vastaamaan joidenkin metrien tarkkuudella referenssilaitteen arvoja sekä polkupyörällä, että autolla testattaessa.

7. YHTEENVETO

Tämän diplomityön tarkoituksena oli suunnitella ja toteuttaa polkupyörän älykäs nopeusmittari. Suunnittelukokonaisuuteen kuuluivat komponenttivalinnat, piirilevysuunnittelu, piirilevyn valmistus ja kokoonpano, kotelon suunnittelu ja valmistus sekä monivaiheinen testaus. Lopputuloksena syntyi toimiva laite, joka täyttää sille asetetut toiminta- ja tarkkuusodotukset.

Komponenttivalinnat tehtiin pitkälti käyttöjännitteen ja saatavuuden perusteella. Kaikkiin tarkoituksiin oli lopulta saatavilla sopiva komponentti, mutta ennen piirilevyn suunnittelua ja valmistusta oli epävarmaa miten erittäin pienien komponenttien kanssa toimiminen onnistuu prototyyppivaiheessa, kun kaikki työ tehtiin käsin. Lopulta piirilevyn valmistus ja kokoonpano onnistuivat erittäin hyvin. Pienimmätkin yksityiskohdat saatiin säilymään levyllä ja juotoksissa ongelmia aiheuttivat ainoastaan täysin koneelliseen valmistukseen tarkoitetut komponentit.

Laitteen kotelo valmistettiin 3D-tulostamalla, josta ei ennen työn suoritusta ollut aiempaa kokemusta. 3D-mallinnus onnistui erittäin helposti, eikä tulostuksen kanssa kohdattu mainittavia ongelmia. Työn edetessä laitteen kotelo muutti kuitenkin muotoaan ilman yksittäistä selvää syytä. Jälkikäteen ajatellen kotelon avoin rakenne olisi pitänyt huomioida suunnittelussa ja järjestää kotelon seinämille tuenta.

Laitteen testausvaiheessa tärkein havainto oli, että kaikki komponentit eivät välttämättä noudata vallitsevia standardeja. GPS-moduulin SPI-väylän poikkeava toiminta aiheutti erillisten lisäkomponenttien tarpeen. Laitteen massavalmistusta ajatellen tämä olisi vaatinut piirilevysuunnittelun uusimisen. Tilannetta olisi kuitenkin ollut käytännössä mahdotonta ennustaa ja testata etukäteen ilman aiempaa kokemusta GPS-moduuleista. Testauksen perusteella opittiin myös, että komponentin datalehdiltä luettavat suoritusarvot ovat vain osa totuutta ja etenkin ohjelmiston suunnittelussa pitää huomioida komponenttien todellinen suorituskyky. Tästä esimerkkinä toimivat ilmanpaineanturin heiluvat mittausarvot.

Ohjelmiston kehityksen edetessä testaamisen tärkeys tuli erittäin hyvin esille. Ohjelmiston suorituskyky on aina kokonaisuus ja keskittyminen yhteen toimintoon muita huomioimatta johtaa helposti poikkeavaan toimintaan. Jatkuva testaaminen kehityksen aikana auttoi työtä etenemään suunnitellun aikataulun mukaisesti.

Vaikka lopputuloksena olikin toimiva laite, tuli suunnittelun aikana ilmi erilaisia kehitys- ja parannusajatuksia, joita ei voitu valmiille laitteelle enää toteuttaa. Tärkein

parannuksen kohde on laitteen fyysinen koko. Kaupalliseksi laitteeksi valmistunut mittari on liian suuri. Kokoa olisi mahdollista pienentää vaihtamalla GPS-moduuli GPS-piiriin ja erilliseen koteloon integroituun antenniin. Myös kaikki 1206-koteloiset vastukset pitäisi korvata pienemmillä versioilla. Lisäksi SD-kortti olisi mahdollista korvata erillisellä muistipiirillä, jolloin laite neuvottelisi tietokoneen kanssa langattomasti tai USB-väylän kautta.

Mietityttämään jäi myös nousu- ja laskumetrien laskenta. Vaikka mittari saatiinkin toimimaan testeissä riittävällä tarkkuudella, havaittiin Garmin Edge 130 referenssilaitteen ja mittarin päivitysnopeudessa eroja. Työssä rakennettu mittari kerryttää metrejä sekunnin välein, kun referenssilaite selvästi suorittaa mittausta useita sekunteja ja päivittää tuloksen sitten näytölle. Laskennassa on selvästi käytössä selvästi laajempi algoritmi.

LÄHTEET

[1] Adesto Technologies, PCB Design and Layout Considerations dor Adesto Momemory Devices, Adesto Application Note AN105-A1, 2018. 23s. Saatavissa: https://www.adestotech.com/wp-content/uploads/PCB_Design_Guidelines.pdf

[2] Atmel, Atmega1281, 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash, Atmel Datasheet, 2014. 435s. Saatavissa: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[3] Elm Chan, FatFs – Generic FAT Filesystem Module, 2018. Saatavissa: http://elm-chan.org/fsw/ff/00index_e.html

[4] Environmental Systems Research Institute, Inc., Understanding Map Projections, 2000, 116s. Saatavissa: http://downloads2.esri.com/support/documentation/ao_/710Understanding_Map_Projections.pdf

[5] Lattice Semiconductor, Improving Noise Immunity for Serial Interface, A Lattice Semiconductor White Paper, 2014, 7s. Saatavissa: www.latticesemi.com/view_document?document_id=50728

[6] Learn EMC, LLC, PCB Layout, EMC Tutorial. Saatavissa: https://learnemc.com/pcb-layout

[7] Limor Fried, ST7565 LCD Library, Adafruit Industries source code, 2010. Saatavissa: https://github.com/adafruit/ST7565-LCD/blob/master/c/stlcd.c

[8] Maestro Wireless Solutions Ltd, GPS Receiver A2235-H A Description of Maestro's GPS Antenna Receiver Module A2235-H. Version 1.3, Maestro Datasheet, 2014. 38 s. Saatavissa: http://update.maestro-wireless.com/GNSS/A2235-H/Maestro_GPS_Receiver_A2235_H_User_Manual_V13.pdf

[9] Microsoft, Microsoft Extensible Firmware Initiative FAT 32 File System Specification, Hardware White Paper version 1.03, 2000, 34s. Saatavissa: http://download.microsoft.com/download/0/8/4/084c452b-b772-4fe5-89bb-a0cbf082286a/fatgen103.doc

[10] National Marine Electronics Association, NMEA 0183 Standard For Interfacing Marine Electronic Devices version 3.01, 2002, 92s.

[11] Nebylov A. V., Aerospace Sensors, Momentum Press, 2013, 349s.

[12] NXP Semiconductors, PCF2129 Accurate RTC with integrated crystal for industrial applications, NXP Datasheet, 2014. 86 s. Saatavissa: https://www.nxp.com/docs/en/data-sheet/PCF2129.pdf

[13] Ott, Henry W., Electromagnetic Compatibility Engineering, Wiley, 2009, 864s.

[14] Russel Jesse, Cohn Ronald, Equirectangular Projection, Book on Demand, 2013, 100s.

[15] SD Group and SD Card Association, SD Specifications Part 1, Physical Layer Simplified Specification Version 6.00, 2018, 262s. Saatavissa: https://www.sdcard.org/downloads/pls/click.php?p=Part1_Physical_Layer_Simplified_Specification_Ver6.00.jpg&f=Part1_Physical_Layer_Simplified_Specification_Ver6.00.pdf&e=EN_SS1

[16] Sitronix, ST7565P 65 x 132 Dot Matrix LCD Controller/Driver, Sitronix Datasheet, 2009. 71 s. Saatavissa: https://www.crystalfontz.com/controllers/Sitronix/ST7565P/299

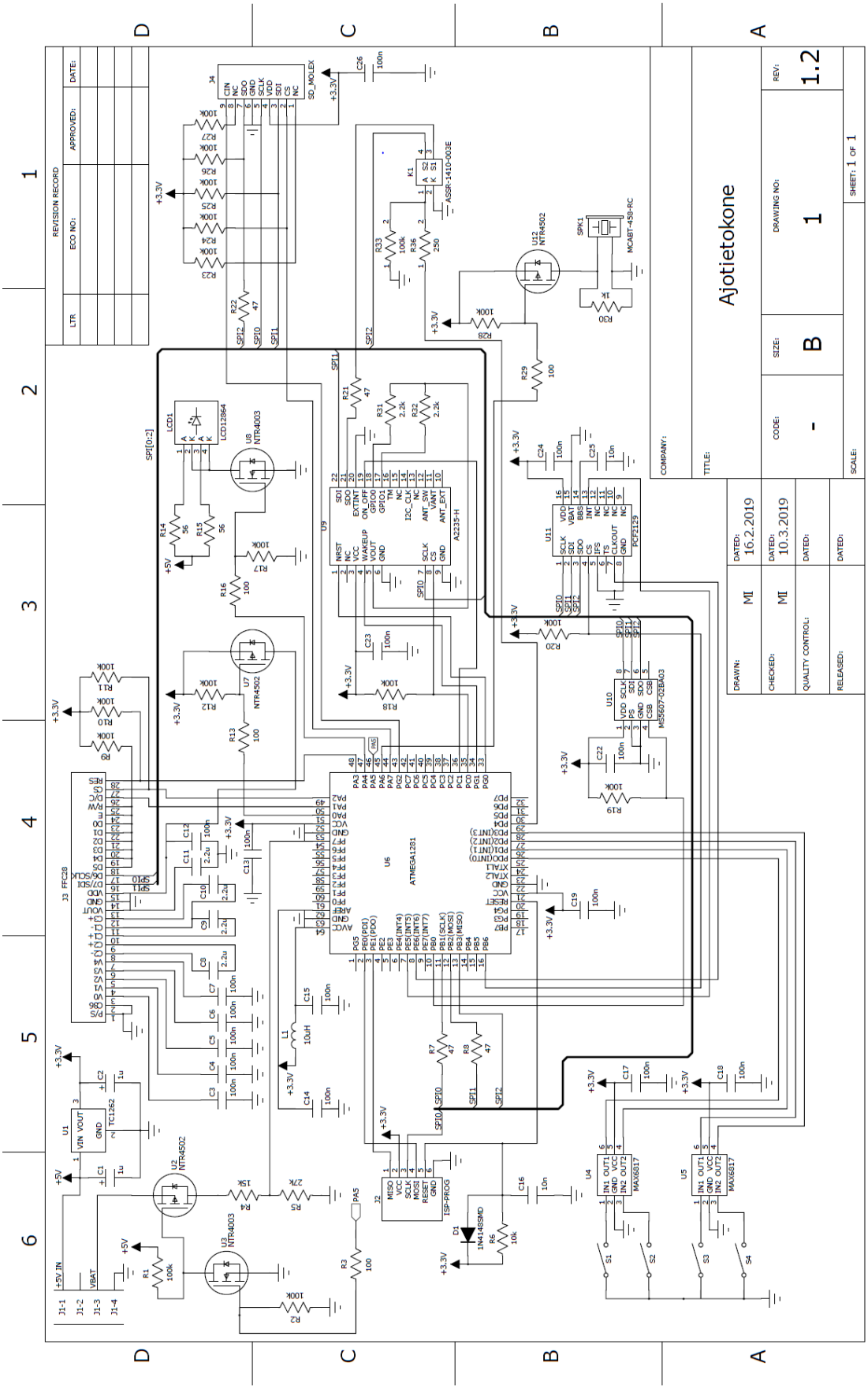
[17] TE connectivity, MS5607-02BA03 Barometric Pressure Sensor, with stainless steel cap, TE datasheet, 2017. 20s. Saatavissa: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5607-02BA03%7FB2%7Fpdf%7FEnglish%7FENG_DS_MS5607-02BA03_B2.pdf%7FCAT-BLPS0035

[18] Wang Guanglong, Dong Yu, Zhu Wenjie, Qiao Zhongtao, Gao Fengqi, High-precision Barometric Altitude Measurement Method and Technology, Proceeding of the IEEE International Conference of Information and Automation, Yinchuan, China, August 2013, 6. Saatavissa: https://ieeexplore.ieee.org/document/6720337

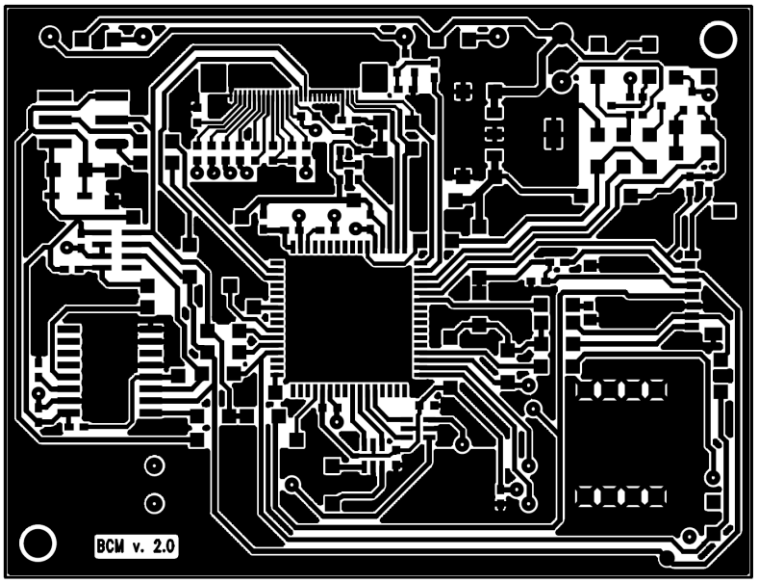
[19] Wikipedia, Design of the FAT file system. Saatavissa: https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system

[20] Xu, Guochang, GPS, Theory, Algorithms and Applications, 2nd Edition, Springer, 2007.

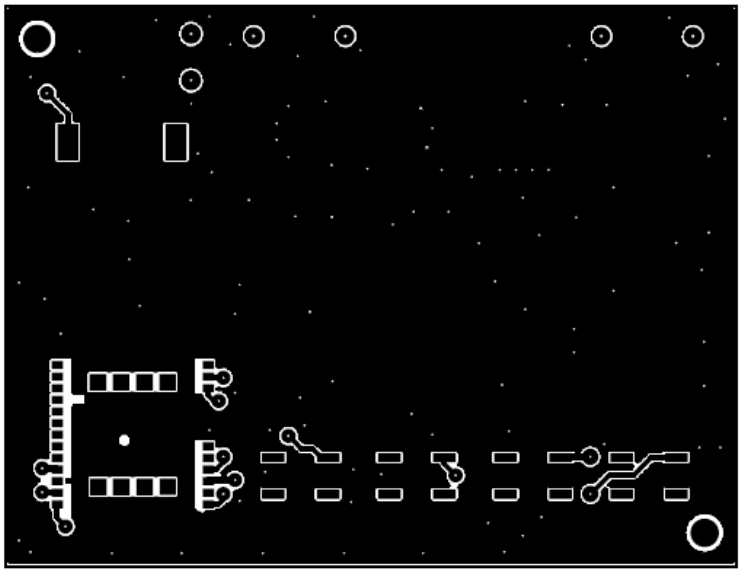
LIITE 1: KYTKENTÄKAAVIO



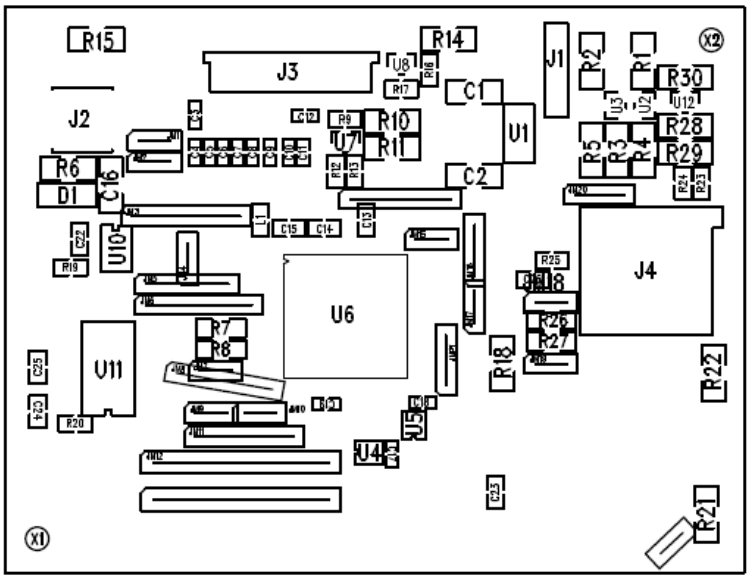
LIITE 2: YLÄPUOLEN VALOTUSMASKI



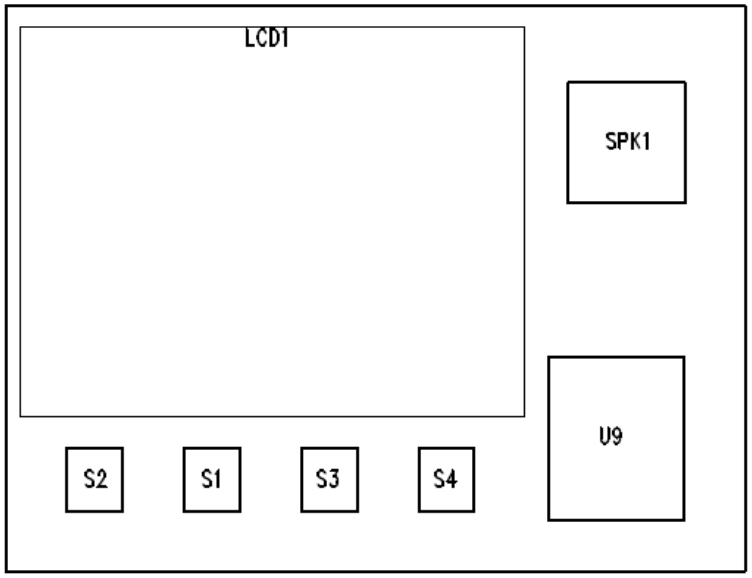
LIITE 3: ALAPUOLEN VALOTUSMASKI



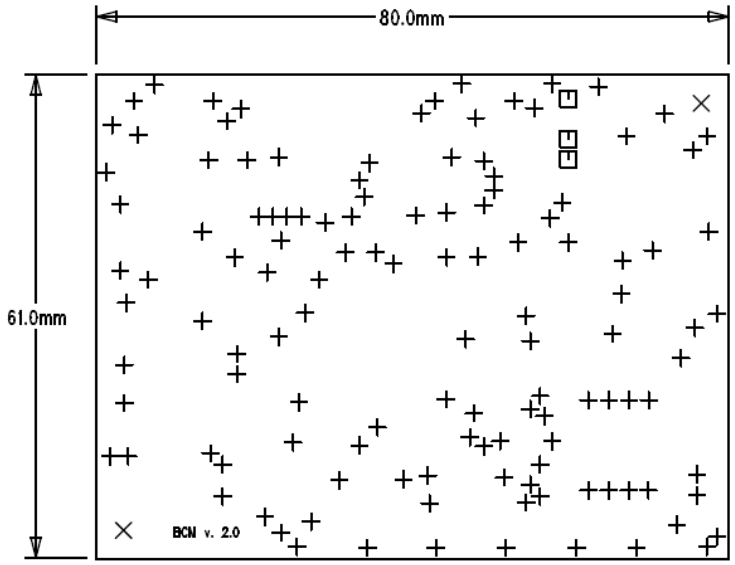
LIITE 4: YLÄPUOLEN OSASIJOITTELUKUVA



LIITE 5: ALAPUOLEN OSASIJOITTELUKUVA



LIITE 6: PORAUSKAAVIO



SIZE	QTY	SYM	PLATED	TOL
0.6	124	+	YES	+/-0.0
3	2	×	YES	+/-0.0
1	3	□	YES	+/-0.0

LIITE 7: TALLENNUSFUNKTIO SD-KORTILLE

```
void save_workout(void) {
    UINT bw;
    uint8_t i,j;

    FIL Fil;        /* File object */

    char buffer[30];        // puskuri muunnoksille

    // Luodaan uusi tiedosto. tiedoston nimeen lisätään ddmmyy jälkeen juokseva numero.
    // Jos tiedosto on jo olemassa, luodaan uusi järjestysnumerolla +1

    cli();
    for(j=0;j<100;++j) {
        // Muodostetaan tiedostonimi. Tyyppi on .txt ja ensimmäiset kuusi merkkiä ddmmyy
        char filename[14];
        sprintf(filename,"%c%c%c%c%c%c_%u.txt",date[0],date[1],date[2],date[3],date[4],date[5],j);

        // Avataan tiedosto kirjoitusta varten
        if (f_open(&Fil, filename, FA_WRITE | FA_CREATE_NEW) == FR_OK) {

            for(i=0;i<30;++i) buffer[i] = ' ';
            // Kirjoitetaan päivämäärä date-muuttujasta
            sprintf(buffer,"Päivämäärä: %c%c.%c%c.%c%c\r\n",date[0],date[1],date[2],date[3],date[4],date[5]);
            f_write(&Fil, buffer, strlen(buffer), &bw);
            for(i=0;i<30;++i) buffer[i] = ' ';
            // Haetaan aika kalenteripiiristä. Aika on BCD-muodossa
            uint8_t data[3],i;
            PORTB &= ~(1 << PB6);                           // kalenteripiirin cs alas
            _delay_us(1);
            spi_transmit(0xA3);
            for(i=0;i<3;++i) data[i] = spi_transmit(0x00);  // luetaan sekunnit [0], minuutit [1] ja tunnit [2]
            PORTB |= (1 << PB6);                            // kalenteripiirin cs ylös
            // Kirjoitetaan aika
            sprintf(buffer,"Aika: %u%uh %u%umin %u%us\r\n",data[2]>>4,data[2]&0x0F,data[1]>>4,data[1]&0x0F,data[0]>>4,data[0]&0x0F);
            f_write(&Fil, buffer, strlen(buffer), &bw);
            for(i=0;i<30;++i) buffer[i] = ' ';
            // Kirjoitetaan kuljettu matka trip-muuttujasta, joka on matka metreinä
            sprintf(buffer,"Matka: %lu,%lu%lu km\r\n",trip/10000,(trip/1000)%10,(trip/100)%10);
            f_write(&Fil, buffer, strlen(buffer), &bw);
            for(i=0;i<30;++i) buffer[i] = ' ';
            // Kirjoitetaan keskinopeus avg_speed-muuttujasta 123 = 12,3 km/h
            sprintf(buffer,"Keskinopeus: %u,%u km/h\r\n",avg_speed/10,avg_speed%10);
            f_write(&Fil, buffer, strlen(buffer), &bw);
            for(i=0;i<30;++i) buffer[i] = ' ';
            // Kirjoitetaan nousumetrit ascent-muuttujasta, joka on nousu senttimetreinä
            sprintf(buffer,"Nousu: %u m\r\n",ascent/100);
            f_write(&Fil, buffer, strlen(buffer), &bw);
            for(i=0;i<30;++i) buffer[i] = ' ';
            // Kirjoitetaan laskumetrit descent-muuttujasta, joka on lasku senttimetreinä
            sprintf(buffer,"Lasku: %u m\r\n",descent/100);
            f_write(&Fil, buffer, strlen(buffer), &bw);

            // Suljetaan tiedosto
            f_close(&Fil);

            // Jos kirjoitus onnistui, poistutaan for-loopista
            break;
        }
    }
    sei();
}
```